Тема 3.5. Основы работы с математическими пакетами

3.5.1. Основные средства и использование математических пакетов
3.5.2. Базовые элементы математического пакета MathCad
3.5.3. Базовые элементы математического пакета MatLab
3.5.4. Контольные вопросы по теме «Основы работы с математическими пакетами (MathCad)»
3.5.5. Контольные вопросы по теме «Основы работы с математическими пакетами (MatLab)»
3.5.6. Тестовые задания по теме «Основы работы с математическими пакетами (MathCad)»
3.5.7. Тестовые задания по теме «Основы работы с математическими пакетами (MatLab)»

3.5.1. Основные средства и использование математических пакетов

Появление компьютеров изменило все сферы современной науки и общественной, и даже личной, жизни. Появилась возможность проводить сложнейшие вычислительные эксперименты, что экономит не только деньги, но и время. Последнее обстоятельство особенно важно для научных работников, педагогов и студентов.

Вместе с тем появление современных систем компьютерной математики позволяет, не отказываясь от принципов фундаментальности классического образования, качественно изменить подходы и методы изложения материала, сделать его более наглядным и доступным, а, следовательно, более интересным и привлекательным для основной массы обучающихся.

Сегодня нечасто вспоминают о том, что компьютеры были созданы в первую очередь для проведения научных расчетов. До сих пор научные и инженерные расчеты остаются одной из важнейших сфер приложения компьютеров. За многие годы накоплены обширные библиотеки научных и учебных подпрограмм, предназначенных для решения типовых задач (задачи линейной алгебры, интегрирование, решение дифференциальных уравнений и других математических задач).

Между тем появление современных систем компьютерной математики позволяет, не отказываясь от принципов фундаментальности классического образования, качественно изменить подходы и методы изложения материала, сделать его более наглядным и доступным, а, следовательно, более интересным и привлекательным для основной массы обучающихся.

В настоящее время появились хорошо работающие математические пакеты, такие как **Maple, Mathematica, Mathcad, Matlab** и некоторые другие. Все упомянутые выше системы, так же как и большинство неупомянутых, пакетов являются весьма дружественными по отношению к пользователю. Конечно же, и синтаксис языка пользователя у них различный, и библиотеки доступных функций могут меняться от нескольких сотен до тысяч, и внутренние структуры и даже используемые алгоритмы значительно отличаются друг от друга, но все они обладают общими свойствами. Таких принципиальных общих свойств значительно больше, чем различий и, таким образом, после освоения одной из систем компьютерной алгебры переход к другой системе не является сложной проблемой.

Для новичка (студента) языки систем компьютерной математики – наиболее простые для использования. Действительно, сначала ему требуется знать лишь несколько функций, которые позволят ему переписать рассматриваемую проблему в виде, очень похожем на ее математическую формулировку. Даже если переписывание выполняется некорректно, интерактивный режим позволяет после нескольких шагов быстро получить результаты,

которые нельзя получить с помощью карандаша и бумаги. А для очень многих приложений этого достаточно.

Пакет **Mathematica**, по-видимому, является сегодня наиболее популярным в научных кругах, особенно среди теоретиков. Пакет предоставляет широкие возможности в проведении символических (аналитических) преобразований, однако требует значительных ресурсов компьютера. Система команд пакета во многом напоминает какой-то язык программирования,

Пакет **Maple** также весьма популярен в научных кругах. Пользователи характеризуют **Maple** как очень надежный и устойчиво работающий **Математический пакет**. Кроме аналитических преобразований пакет в состоянии решать задачи численными методами. Характерной особенностью пакета является то, что ряд других программных продуктов используют интегрированный символический процессор **Maple**.

Подобно упомянутым выше пакетам, пакет **Matlab** фактически представляет собой своеобразный язык программирования высокого уровня, ориентированный на решение научных и учебных задач. Характерной особенностью пакета является то, что он позволяет сохранять документы в формате языка программирования **C**.

Пакет **Mathcad** популярен, пожалуй, более в инженерной и учебной и учебной среде. Характерной особенностью пакета является использование привычных стандартных математических обозначений, то есть документ на экране выглядит точно так же обычный математический расчет. Для использования пакета не требуется изучать какую-либо систему команд, как, например, в случае пакетов **MatLaB** или **Maple**. Пакет ориентирован в первую очередь на проведение численных расчетов, но имеет встроенный символический процессор **Maple**, что позволяет выполнять аналитические преобразования. В последних версиях предусмотрена возможность создавать связки документов **Mathcad** с документами **Matlab**. В отличие от упомянутых выше пакетов, **Mathcad** является средой визуального программирования, то есть не требует знания специфического набора команд. Простота освоения пакета, дружественный интерфейс, относительная непритязательность к возможностям компьютера явились главными причинами того, что именно этот пакет стал наиболее популярным при обучении студентов. Однако, в отличие от алгоритмических языков программирования, в которых синтаксические тонкости требуют тщательного изучения, в то время как принципы работы компилятора можно полностью игнорировать, здесь пользователь должен разобираться, «как это работает», в частности, как представляются и обрабатываются данные.

В действительности, хотя обычно трудно предсказать время вычисления и размер результатов, знание принципов работы может дать представление о порядке их величины и при необходимости оптимизировать их. Эти оценки в действительности существенны: для большинства математических вычислений результаты получаются почти моментально, и все идет отлично. Но если это не так, то требуемое время и память возрастают обычно экспоненциально. Таким образом, выполнимость данных вычислений не всегда очевидна, и глупо жертвовать значительными ресурсами, когда неудачу можно предсказать заранее.

Поэтому владение эффективным стилем программирования и способность предвидеть размер вычислений являются здесь значительно более весомыми, чем в численных расчетах, где возрастание обычно бывает линейным. К сожалению, это в значительной степени приобретается с опытом.

В последнее время просматривается тенденция к сближению и интеграции различных пакетов. Например, последние выпуски пакетов **Mathematica** и **Maple** имеют хорошие возможности для визуального программирования; в **Matlab** включена библиотека аналитических преобразований **Maple**; **Mathcad** позволяет работать совместно с **Matlab**.

К сожалению, существует настоящая пропасть между теми численными методами, которые описаны в учебниках,м учебных пособиях для студентов и теми, которые применяются на практике. В замечательной, хотя и недоступной для большинства студентов книге

«Numerical Recipes in **C**», авторы замечают: «Увы, времена меняются; ... классические формулы почти абсолютно бесполезны. Они являются музейными экспонатами, хотя и прекрасными», В данной статье делается попытка перебросить мостик через эту пропасть.

Обычно, начиная работать с любой из математических пакетов, студент достаточно легко решает небольшие и несложные примеры и задачи из учебника. Однако, приступая к решению настоящих (реальных) задач, пользователь сталкивается с рядом проблем: то компьютер слишком долго считает, то не хватает памяти, то в ответе получается формула на 5-10 страниц, а то машина выдает и вообще неправильный ответ. После этого встает вопрос – «Стоит ли тратить время на детальное изучение таких "игрушечных" систем и не лучше ли потратить это время на написание самих формул?».

Бездумное применение пользователем математических пакетов таит в себе большие проблемы.

Необходимо отметить, что пользователи пакетов компьютерной математики должны иметь представление об основных численных методах. Вообще говоря, появление современных вычислительных систем значительно облегчает доступ к компьютеру непрофессионалам в области программирования, и поддерживает постоянное стремление к их усовершенствованию и освоению новых компьютерных технологий.

Системы компьютерной математики представлены разработками различных фирм (**MathSoft, MathWorks, Maple, Wolfram** и др.). Прежде чем начать изучение конкретных систем, оценивая их достоинства и недостатки, мы познакомимся со структурой, принципами работы и элементами, которые характерны для всех систем компьютерной математики.

Довольно условно структура **СКМ** показана на рис. 5.1.1-1.

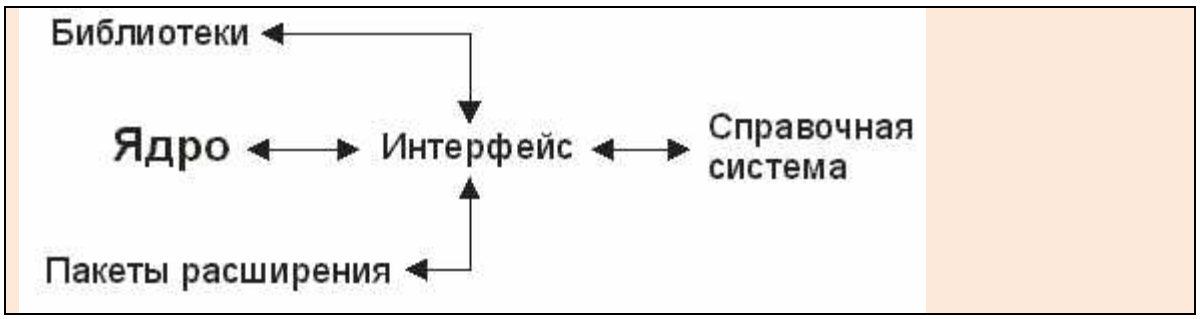


Рис. 5.1.1-1. Структура универсальных систем компьютерной математики

Ядро системы содержит коды множества быстро исполняемых функций и процедур, обеспечивающих достаточно представительный набор встроенных функций и операторов системы. Их число в ядре современных СКМ может достигать многих тысяч. Например, ядро системы Mathematica содержит данные о более чем 5000 одних только интегралов, хотя для интегрирования используются только несколько встроенных функций.

Интерфейс современных СКМ характерный для всех **Windows**-приложений, обеспечивает присущие им удобства работы и дает пользователю возможность обращаться к ядру со своими запросами и получать результат решения на экране.

Библиотеки содержат процедуры и функции, которые используются более редко. Это связано с тем, что функции и процедуры, включенные в ядро, выполняются быстро, если их не слишком много, и поэтому объем ядра ограничивают Общее число доступных пользователю функций ядра и библиотек достигает нескольких тысяч.

Пакеты расширения кардинально расширяют возможностей систем и их адаптацию к решаемым конкретными пользователями. Эти пакеты (нередко и библиотеки) пишутся на собственном языке программирования той или иной СКМ, что делает возможным их подготовку обычными пользователями. Наращивание возможностей систем с помощью пакетов расширения практически ничем не ограничено.

Справочная система стала нормой для сопровождения компьютерных математических систем. Справочная система обычно поддерживает следующие возможности доступа к справочным данным: оперативная; всплывающая подсказка по элементам интерфейса, получаемая наведением на них указателя мыши; оперативная справка по операторам и функциям, получаемая нажатием клавиши F1 при курсоре ввода, установленном на операторе или в имени функции; оперативная справка, получаемая вводом символа ? или слова help, после которого указывается имя объекта, по которому требуется справка; и др.

Рекордсменом по обилию справочных материалов является система **MatLab**. Объем только описаний системы в формате файлов RTF достигает более200 Мбайт – это соответствует десяткам книг обычного формата. По существу, с системой поставляется уникальная справочная информация по всем вопросам применения математики. И эта электронная документация является лишь частью полных справочных материалов. В их числе сотни эффективных примеров применения системы. Здесь особо надо отметить систему Maple – в ее справочной системе около десятка тысяч примеров.

К сожалению, справочные системы англоязычные, что резко снижает их ценность для русскоязычных пользователей. Тем не менее, именно справочные системы содержат детальное описание интерфейса, операторов и функций, которое трудно найти в книгах и руководствах пользователя.

Необходимо отметить, что ядро, библиотеки, пакеты расширения и справочная система современных СКМ аккумулируют знания в области математики, накопленные за тысячелетия ее развития. Поэтому СКМ относят к *интеллектуальным* программным продуктам, одно из назначений которых – предоставление пользователю знаний в области численных методов расчета и моделирования, аналитической математики и современной графики.

3.5.2.Базовые элементы математического пакета MathCad

3.5.2.1. Рабочая среда MathCad и простейшие вычисления

Mathcad является уникальной системой для научных и инженерных расчетов и позволяет работать с формулами, числами, текстом и графиками. С помощью Mathcad можно решить почти любую математическую задачу аналитически либо численно. Mathcad позволяет записывать на экране компьютера формулы в их привычном виде.

Mathcad имеет свою собственную справочную систему. Электронные Книги делают доступными для использования в рабочем документе множество полезных формул, справочных данных и диаграмм простым нажатием кнопки.

Объединяя в одном рабочем листе текст, графику, и математические выкладки, Mathcad облегчает понимание самых сложных вычислений.

Интерфейс Mathcad аналогичен интерфейсу других Windows-приложений. После запуска пакета на экране возникает рабочее окно (рис.3.5.2-1) со следующими панелями:

- панель Главного меню (рис.3.5.2-2);
- панель Стандартная, содержащая основные команды и операции (рис.5.5.2-3);

- панель Форматирование, содержащая операции по выбору типа и размера шрифтов, расположения текста и т.п.;
- панель Математика, содержащая кнопки с палитрами часто используемых математических обозначений (рис.3.5.2-4);
- нижняя строка – строка состояния, где приводится информация о текущих режимах.
- линейки вертикальной и горизонтальной прокрутки, используемые при работе с большими документами.

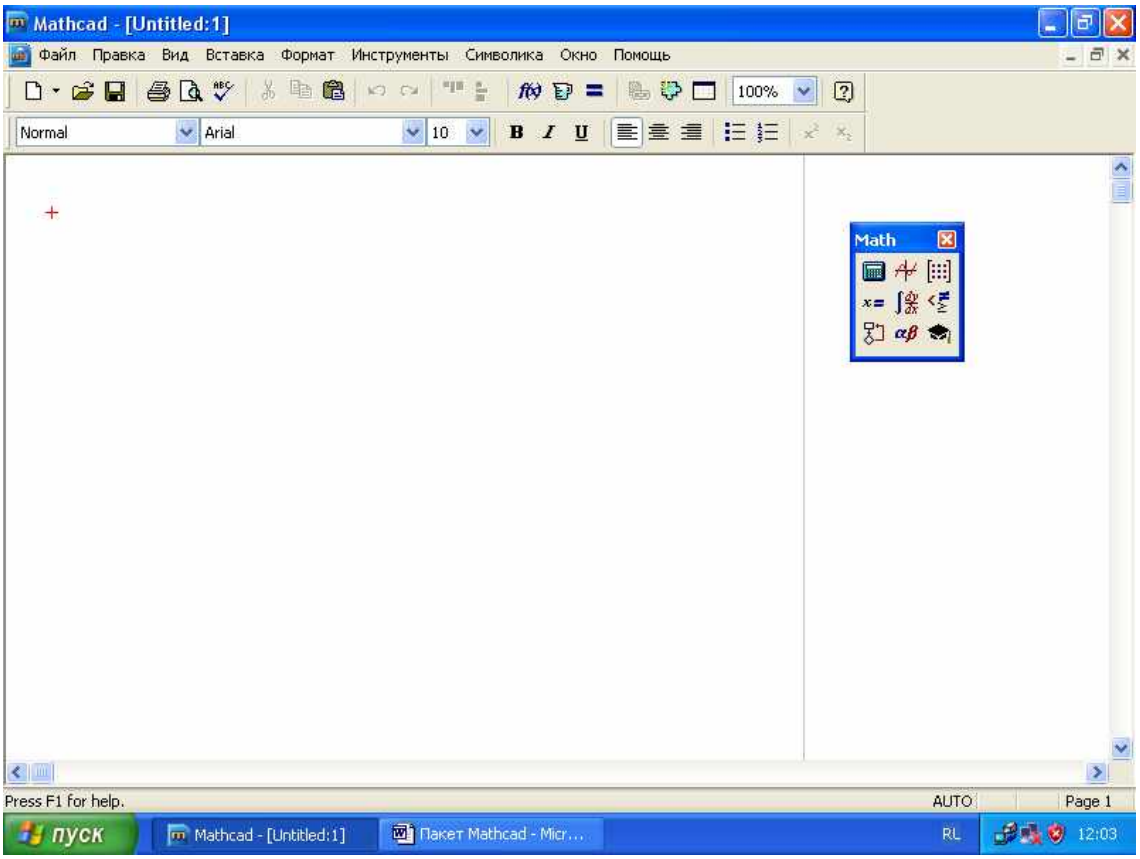
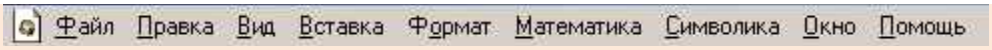


Рис. 3.5.2-1. Вид рабочего окна после первой загрузки



1. **Кнопка раскрытия системного меню Mathcad.**
2. **Команды, связанные с созданием, открытием, сохранением и печатью файлов.**
3. **Команды, связанные с правкой текста.**
4. **Команды, управляющие внешним видом документа в рабочем окне.**
5. **Команды вставки различных объектов в документ.**
6. **Команды форматирования текста, формул и графиков.**
7. **Команды управления вычислительным процессом.**
8. **Команды символьных вычислений.**
9. **Команды расположения окон документов на экране.**
10. **Вызов справочной информации.**

Рис. 3.5.2-2. Главное меню

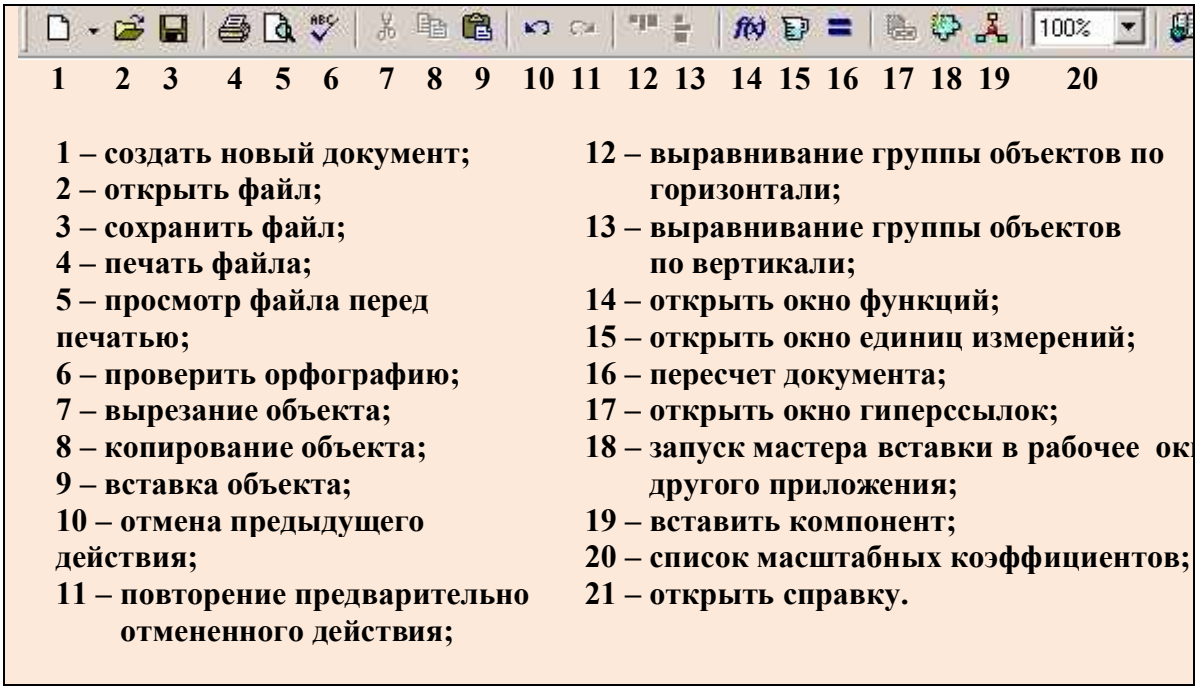


Рис. 3.5.2-3. Назначение кнопок панели инструментов Стандартная

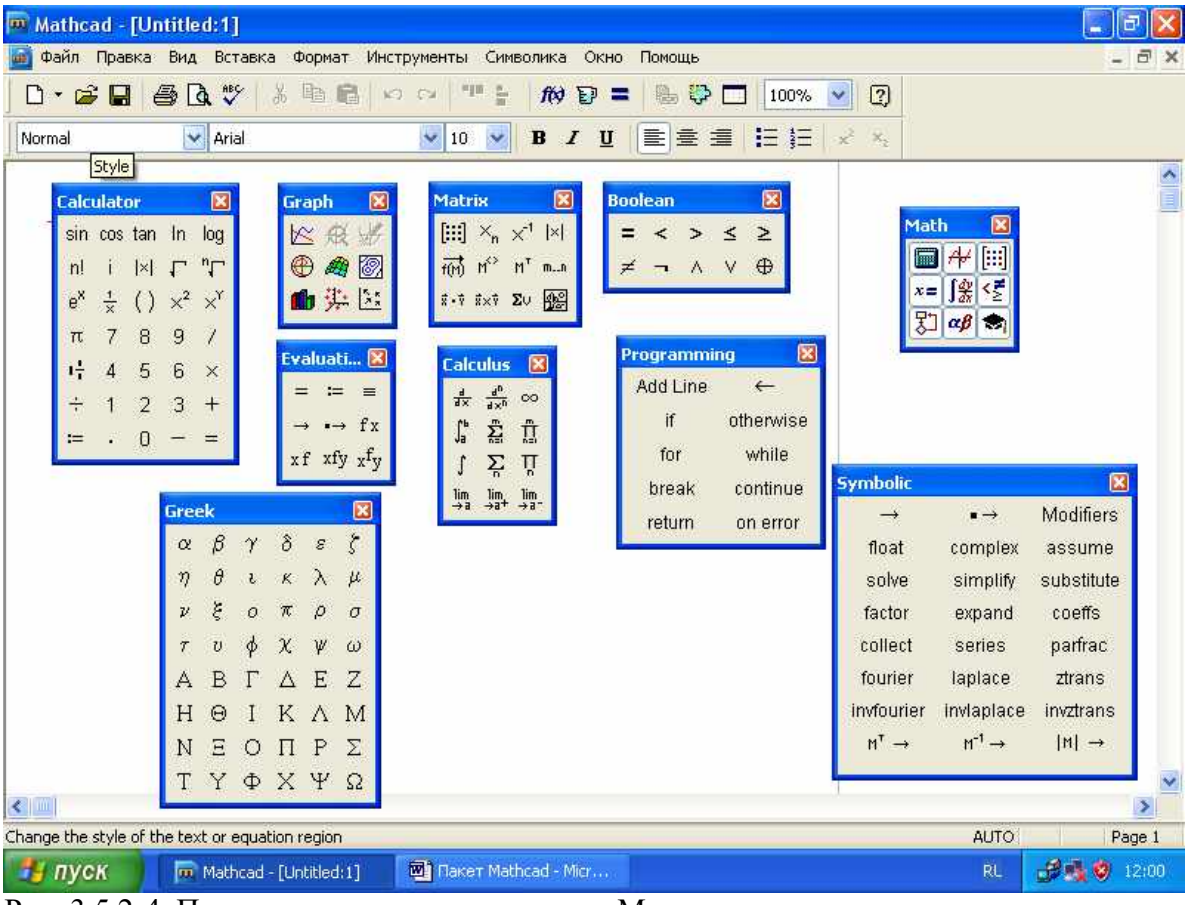


Рис. 3.5.2-4. Палитры панели инструментов Математика

Рассмотрим **«Основные объекты системы Mathcad».**

Общение с пользователем системы Mathcad осуществляется с помощью математически ориентированного входного языка, который является типичным языком визуального программирования. Большинство операторов и функций входного языка

знакомо пользователю по курсу математики. Благодаря этому большая часть расчетов в Mathcad не требует программирования в общепринятом смысле этого слова.

Алфавит входного языка – это совокупность символов и слов, которые используются при задании команд и функций, необходимых для решения интересующего пользователя класса задач. Алфавит системы Mathcad содержит строчные и прописные буквы латинского алфавита, цифры, ряд греческих букв и специальных знаков.

Общение пользователя с системой происходит на некотором промежуточном математически ориентированном языке визуального программирования. Последовательность математических вычислений проводится с использованием операторов и функций. Кроме того, к алфавиту Mathcad также относятся, все знаки, которые можно ввести с помощью палитр панели Математика.

Число – простейший объект языка Mathcad, представляющий количественные данные. В Mathcad числа могут быть целыми, дробными, с фиксированной и плавающей точкой (в экспоненциальном виде). Например: 9, -65, 0.0005, 76.e-4, -5.67e12.

Для представлений комплексных чисел, имеющих действительную и мнимую части, в Mathcad используется переменная (i или j), служащая множителем мнимой части и означающая корень квадратный из -1. Например: 7i, -5.4+i7, 0.004e-j0.1.

Операторы – элементы языка, предназначенные для создания математических выражений совместно с данными, именуемыми операндами. Наиболее известны арифметические операторы, например, сложение (+), вычитание(-), умножение (*) и деление (/).

Функции – это объекты входного языка, имеющие имя и параметры, указываемые в круглых скобках через запятую. Имя функции, как правило, соответствует имени математической функции. Функции обладают свойством возвращать некоторое значение, в ответ на обращение к ним по имени с указанием аргумента или списка аргументов. Полный перечень встроенных функций Mathcad насчитывает более 290 позиций. Поэтому приведём лишь некоторые из них, отметив, что некоторые специальные функции, предназначенные для реализации численных методов, будут подробно рассмотрены в Разделе 6:

atan(z) – арктангенс;
bsplin(vx,vy,u,n) – вектор коэффициентов B-сплайна степени n (1, 2 и 3) для данных, представленных векторами vx и vy, и вектора u, имеющего (n-1) элементов;
ceil(x) – наименьшее целое, не превышающее x;
cols(A) – число столбцов в матрице A;
combin(n,k) – возвращает число сочетаний k из n>k;
corr(vx,vy) – коэффициент корреляции векторов vx и vy;
cos(z) – косинус;
cot(z) – котангенс;
exp(z) – значение e (основание натурального логарифма) в степени z;
Find(var1, var2,…) - значение var1, var2,…, дающее решение системе уравнений в блоке, объявленном Given , который может содержать условия ограничения;
floor(x) – наибольшее целое число, меньшее или равное x;
Im(z) – мнимая часть комплексного числа z;
ln(z) – натуральный логарифм;
log(z) – десятичный логарифм;
max(M) – наибольший элемент в матрице A;
min(A) – наименьший элемент в матрице A;

minerr(x1,x2,…) – вектор значений для x1,x2,…, которые приводят к минимальной ошибке в системе уравнений;
mod(x, modulus) – остаток от деления x по модулю (аргументы - действительные числа, результат имеет такой же знак, как и x);
rank(A) – ранг квадратной матрицы A;
Re(z) – действительная часть комплексного числа;
rnd(x) – псевдослучайное число с равномерным распределением в интервале [0, x];
root(expr, var) – значение переменной var, при которой выражение expr равно нулю (в пределах точности TOL);
sin(z) – синус;
sort(v) – вектор v, отсортированный по убыванию;
tan(z) – тангенс.

Операторы и функции используются для создания математических выражений.

Данные, используемые для проведения вычислений, могут быть представлены числовыми константами и переменными. Наиболее часто встречающиеся встроенные константы Mathcad представлены в таблице 3.5.2-1.

Таблица 3.5.2-1

Переменная	Знач. по умолчанию	Назначение и использование
π	3.1415926535897931	**Число π из 17 десятичных цифр**
e	2.7182818284590451	**Число e из 17 десятичных цифр**
∞	10^{307}	**Бесконечность представляется максимальным числом в среде пакета**
TOL	**0.001**	**Величина ошибки, используемая при приближенных вычислениях**
i или j		**Мнимая единица**
ORIGIN	**0**	**Нижняя граница индекса массивов**

Имена переменных формируются по общепринятому принципу –- уникальное имя, начинающееся с буквы. Для присваивания переменной значения используется следующая конструкция: x:= 5. Символ присваивания (:=) отличен от математического знака равенства (=) и вводится с клавиатуры символом двоеточия или палитры Калькулятор панели Математика. Отметим, что не следует применять переменную в выражениях до присваивания этой переменной конкретного значения.

Перед началом работы курсор на экране имеет вид крестика, в момент ввода информации приобретает вид синего уголка.

Ниже на рис. 3.5.2-5 приведены примеры построения математических выражений, где переменным присвоены числовые значения, использованы встроенные (стандартные) функции, выведены значения констант (e и π), а также вычислены значения выражений, использующих переменные и константы. Для проведения вычислений использован знак равенства (=).

Выражения	Стандартные функции	Числовые константы
$x := 5 \quad y := 3 \quad a := 10$ $5 + 8 = 13 \quad x + y = 8$ $\frac{(x+y)}{4} = 2$	$\ln(x) = 1.609 \quad \sin(x) = -0.959$ $\tan(a) = 0.648 \quad \sinh(x) = 74.203$ $\int_0^2 x^2\,dx = 2.667 \quad \frac{d}{dx}x^3 = 75$	$e = 2.718$ $\pi = 3.142$

Рис. 3.5.2-5

Удобство и эффективность расчетов в Mathcad прежде всего определяется возможностью создания и использования функций пользователя. Они позволяют проводить многократные вычисления одного и того же выражения. Имя функции образуется по тем же правилам, что и имя переменной, но здесь следом за именем в скобках (через запятую) перечисляются параметры. Перед использованием функции в вычислениях значения всех ее параметров и переменных, входящих в ее выражение, должны быть определены. Например,

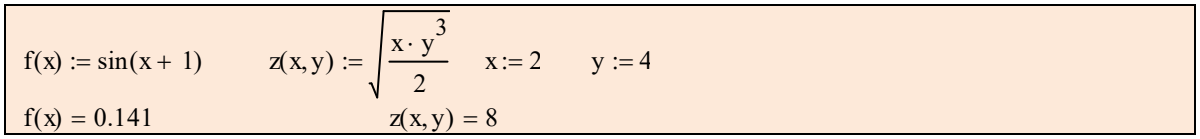


Рис. 3.5.2-6

Mathcad не делает различий между именами переменных и функций. Это означает, что, если функцию определить как f(x), то в дальнейших расчетах нельзя использовать переменную с именем f.

Одним из важнейших понятий системы Mathcad является дискретная переменная. Под дискретной переменной понимается ряд чисел, выстроенных в порядке возрастания или убывания, поэтому её иногда называют интервальной переменной. С помощью дискретных переменных можно задать как целые, так и дробные значения переменной, но обязательно равноотстоящие друг от друга. Пример определения дискретных переменных показан на рис. 3.5.2-7.

Рис. 3.5.2-7

Это означает, что переменная x принимает значения от 0 до 5 с шагом 1, а переменная a – от 1 до 2 с шагом 0.1, другими словами, при описании дискретных переменных указывается первое, второе и последнее значения. Если второе значение переменной опущено, то это означает, что шаг ее изменения равен 1.

Знак диапазона (две точки) нельзя набирать с клавиатуры, нажимая два раза клавишу «.» (точка). Вместо этого надо использовать клавишу «;» (точка с запятой).

Дискретная переменная практически играет роль оператора цикла, что позволяет получить таблицу значений функции для ряда значений переменных и построить график функции на множестве значений переменной.

Перед построением таблицы функцию надо описать, а всем переменным, входящим в ее состав, должны быть присвоены числовые значения. Диапазон и шаг изменения аргумента следует задать с использованием дискретной переменной. Теперь, если после имени функции ввести знак равенства, на экране появится таблица значений функции в заданном диапазоне изменения аргумента. Перед выводом таблицы значений функции рекомендуется выводить соответствующие им значения аргумента, таблица которых выводится аналогично (имя аргумента и знак равенства). В приведенном ниже примере на рис. 3.5.2-8 a и b – переменные, а x – параметр функции y(x).

$a := 4 \qquad b := 3.4$

$$y(x) := a + \frac{\sin\left(x^2\right)}{b}$$

$x := 1, 1.2 .. 2$

x =	y(x) =
1	4.247
1.2	4.292
1.4	4.272
1.6	4.162
1.8	3.971
2	3.777

Рис. 3.5.2-8

Вычисления выражений в Mathcad проводятся с точностью, соответствующей 20 знакам, но на экран выводятся не все значащие цифры (по умолчанию 3). Чтобы изменить формат вывода числового результата, достаточно установить курсор на нужном числовом результате, дважды щелкнуть мышкой, и в появившемся окне провести форматирование. На рис.3.5.2-9 окно Формат Результата открыто на вкладке Формат чисел, где перечислены доступные форматы.

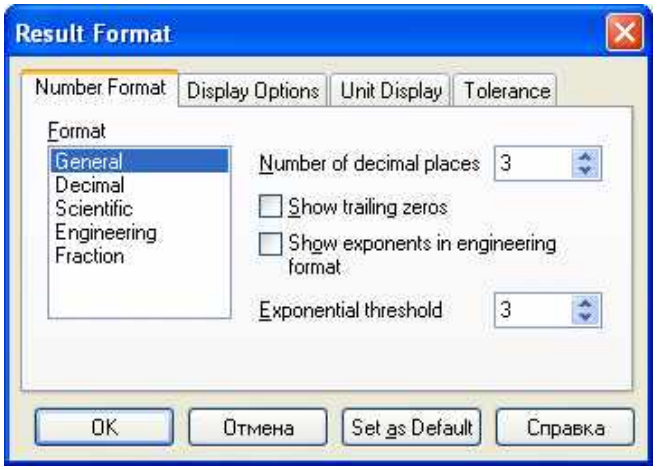


Рис. 3.5.2-9

General (Главный) – формат принят по умолчанию, числа отображаются с порядком, а число знаков мантиссы определяется параметром Экспоненциальный порог.

Decimal (Десятичный) – формат выводит результат в десятичном представлении чисел с плавающей точкой (например, 12.256).

Scientific (Научный) – формат отображает числа только с порядком (например, $1.22*10^5$).

Engineering (Инженерный) – формат отображает числа только с порядком, кратным 3 (например, $1.22*10^6$).

Fraction (Дробный) – формат отображает числа в виде правильной или неправильной дроби (например, $\frac{5}{3}$ или $1\frac{2}{3}$).

Кроме вида формата, в окне можно изменить число знаков после десятичной точки и число знаков, выводимое после десятичной точки (порог экспоненциального представления). Выбранные параметры могут быть применены только к выделенному числу (кнопка OK) или ко всему документу (кнопка Set as Default). Ниже на рис. 3.5.2-10 приведены примеры отображения результатов в различных форматах отображения.

$a := e^{10}$	Главный (по умолчанию)
$a = 2.203 \times 10^{4}$	Научный
$a = 22026.466$	Десятичный
$a = 22.026 \times 10^{3}$	Инженерный
$a = \frac{11053739568}{501839}$	Дробный

Рис. 3.5.2-10

Работа с векторами и матрицами.

Определение вектора или матрицы в Mathcad производится с помощью кнопки Матрица панели Математика. Для этого необходимо, установив курсор на место ввода, ввести имя матрицы (имена матриц в математике принято отображать большими буквами) и оператор присваивания, затем щелкнуть по изображению шаблона матрицы. На экране появится диалоговое окно (рис.3.5.2-11), в которое надо ввести число строк и столбцов матрицы и щелкнуть по кнопке OK. На экране появится шаблон матрицы.

Рис.3.5.2-11

С помощью шаблона можно ввести матрицу, содержащую не более 100 элементов. Доступ к любому элементу матрицы можно получить через имя матрицы с двумя индексами. Первый индекс обозначает номер строки, а второй – номер столбца.

Вектор – это матрица, состоящая из одного столбца, поэтому произвольный элемент вектора задается одним индексом.

Для набора нижнего индекса удобно пользоваться клавишей [(открывающая квадратная скобка). Нумерация элементов массива (вектора или матрицы) может начинаться с 0, 1 или любого другого числа (положительного или отрицательного). Установкой начального индекса управляет встроенная функция ORIGIN. По умолчанию ORIGIN=0. Это означает, что первый член массива имеет номер 0. Чтобы, как обычно принято в математике, нумерация начиналась с 1, надо перед вводом матрицы набрать строчку: ORIGIN=1.

Mathcad позволяет выполнить над матрицами основные арифметические действия, включая сложение, вычитание и умножение, а также операции обращения, вычитания определителя матрицы, нахождения собственных чисел и собственных векторов и т.д. Основные действия над матрицами нашли свое отражение на палитре Матрица. Примеры численного и символьного выполнения этих операций приведены на рис. 3.5.2-12.

$$\text{ORIGIN} = 0 \qquad i := 0..2 \qquad j := 0..2$$

$$D_{i,j} := 10 - i - j$$

$$D = \begin{pmatrix} 10 & 9 & 8 \\ 9 & 8 & 7 \\ 8 & 7 & 6 \end{pmatrix} \qquad D := D^{T} \qquad D = \begin{pmatrix} 10 & 9 & 8 \\ 9 & 8 & 7 \\ 8 & 7 & 6 \end{pmatrix} \qquad D_{0,0} = 10$$

$$\text{ORIGIN} := 1 \qquad D_{1,1} = 10$$

$$B := \begin{pmatrix} 3 & 4 & 5 \\ 4 & 5 & 1 \\ 1 & 2 & 8 \end{pmatrix} \quad B + D = \begin{pmatrix} 13 & 13 & 13 \\ 13 & 13 & 8 \\ 9 & 9 & 14 \end{pmatrix} \quad B - D = \begin{pmatrix} -7 & -5 & -3 \\ -5 & -3 & -6 \\ -7 & -5 & 2 \end{pmatrix}$$

$$B \cdot D^T = \begin{pmatrix} 106 & 94 & 82 \\ 93 & 83 & 73 \\ 92 & 81 & 70 \end{pmatrix} \quad B \cdot B^T = \begin{pmatrix} 50 & 37 & 51 \\ 37 & 42 & 22 \\ 51 & 22 & 69 \end{pmatrix} \quad |B| = 5$$

$$B^{-1} = \begin{pmatrix} 7.6 & -4.4 & -4.2 \\ -6.2 & 3.8 & 3.4 \\ 0.6 & -0.4 & -0.2 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot (1 \ 2 \ 3) = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix} \quad (1 \ 2 \ 3) \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = (14)$$

Рис. 3.5.2-12

Для работы с векторами и матрицами в системе Mathcad имеется ряд встроенных векторных и матричных функций (табл. 3.5.2-2), делающих работу с векторами и матрицами столь же простой, что и с обычными числами и переменными.

Таблица 3.5.2-2

Функция	Назначение
length(V)	Возвращает число элементов вектора
last(V)	Возвращает номер последнего элемента
max(V)	возвращает максимальный элемент вектора или матрицы
min(V)	возвращает минимальный элемент вектора или матрицы
Re(V)	возвращает вектор действительных частей комп. вектора
Im(V)	возвращает вектор мнимых частей вектора с комплексными эл.
augment(M1,M2)	объединяет в одну матрицы M1 и M2, имеющие одинаковое число строк (объединение идет бок о бок), присоединяя первый столбец M2 к последнему столбцу M1.
stact(M1,M2)	объединяет матрицы M1 и M2, имеющие одинаковое число столбцов, сажая M1 над M2
diag(V)	создает диагональную матрицу, элементы главной диагонали которой – элементы вектора V
cols(M)	возвращает число столбцов матрицы M
rows(M)	возвращает число строк матрицы M
rank(M)	возвращает ранг матрицы M
mean(M)	возвращает среднее арифметическое элементов массива M

Ниже на рис. 3.5.2-13 приведены примеры наиболее распространенных векторных и матричных операторов.

$$B := \begin{pmatrix} 1 \\ 2 \\ 12 \end{pmatrix} \quad A := \begin{pmatrix} 5.6 & 8 & 3.6 & 9 \\ 9 & 5 & -11 & 10 \\ -12 & 8 & 3 & 0 \end{pmatrix} \quad C := \begin{pmatrix} 1 & 3 & 7 \\ 4 & 1 & 9 \\ 5 & 1 & 1 \end{pmatrix} \quad V := \begin{pmatrix} 1 \\ 3 \\ 6 \end{pmatrix}$$

$$\text{length}(B) = 3 \quad \max(A) = 10 \quad \text{diag}(V) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 6 \end{pmatrix}$$

$$\min(A) = -12 \quad \text{last}(B) = 2 \quad \text{cols}(A) = 4$$

$$\text{rows}(A) = 3 \quad \text{rank}(C) = 3 \quad \text{mean}(C) = 3.556$$

Рис. 3.5.2-13.

Кроме набора математических функций, при работе с матрицами большое удобство представляет наличие в Matcad функций сортировки – перестановки элементов векторов и матриц:

sort(V) – сортировка элементов вектора в порядке возрастания их значений;
reverse(V) – сортировка элементов вектора в порядке убывания их значений;
csort(M,n) – перестановка строк матрицы M таким образом, чтобы отсортированным оказался n–й столбец;
rsort(M,n) – перестановка строк матрицы M таким образом, чтобы отсортированной оказалась n–я строка.

Использование перечисленных функций можно проиллюстрировать следующими примерами, приведенными на рис. 3.5.2-15.

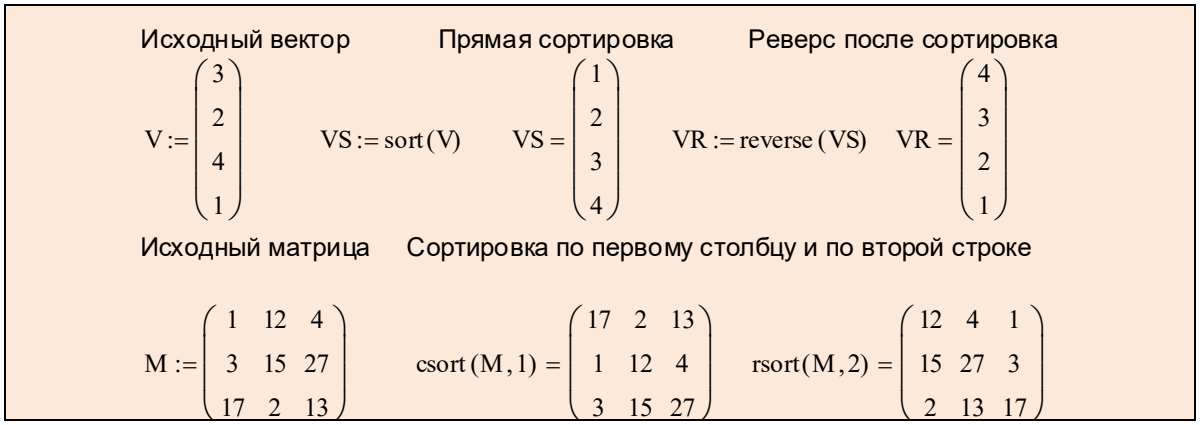


Рис. 3.5.2-14

3.5.2.2. Ввод и редактирование формул

Формульный редактор Mathcad позволяет быстро и эффективно вводить и изменять математические выражения. Тем не менее, некоторые аспекты его применения не совсем тривиальны. Предварительное изучение особенностей редактора формул позволяет пользователю значительно сэкономить время при реальной работе в пакете Mathcad.

К основным элементам интерфейса редактора формул Mathcad относятся:

- указатель мыши, играющий обычную для приложений Windows роль;
- курсор - обязательно находящийся внутри документа в одном из трех видов:
  - - курсор ввода - это крестик красного цвета, который отмечает пустое место в документе, куда можно вводить текст или формулу;
  - - линии ввода - горизонтальная и вертикальная линии синего цвета, выделяющие в тексте или формуле определенную часть;
  - - линия ввода текста - красная вертикальная линия, аналог линий ввода для текстовых областей;
- местозаполнители - появляются внутри незавершенных формул в местах, которые должны быть заполнены символом или оператором:
  - - местозаполнитель символа - черный прямоугольник;
  - - местозаполнитель оператора — черная прямоугольная рамка.

Виды курсоров и местозаполнителей, относящиеся к редактированию формул, представлены на рис. 3.5.2-15.

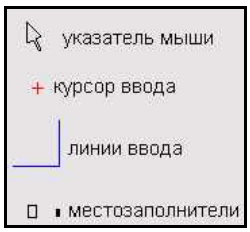


Рис. 3.5.2-15. Элементы интерфейса редактирования

Большую часть окна Mathcad занимает рабочая область документа Mathcad, в которую пользователь вводит математические выражения, текстовые поля и элементы программирования. Ввести математическое выражение можно в любом пустом месте документа Mathcad. Для этого следует поместить курсор ввода в желаемое место документа, щелкнуть в нем мышью, и просто начать вводить формулу, нажимая клавиши на клавиатуре. При этом в документе создается математическая область, которая предназначена для хранения формул, интерпретируемых процессором Mathcad (рис. 3.5.2-16).

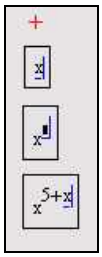


Рис. 3.5.2-16. Пример посимвольного ввода формулы

Поместить формулу в документ можно и просто начиная вводить символы, числа или операторы, например + или /, входящие в формулу. В этом случае на месте курсора ввода также создается математическая область, иначе называемая регионом, с формулой и линиями ввода, но в зависимости от типа оператора автоматически появляются и местозаполнители, без заполнения которых формула не будет восприниматься процессором Mathcad (рис. 3.5.2-17).

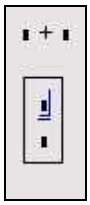


Рис. 3.5.2-17. Пример начала ввода формул с операторов

Редактировать формулы в Mathcad можно так, как подсказывают вам интуиция и опыт работы с другими текстовыми редакторами. Большинство операций правки формул реализованы естественным образом, однако некоторые из них несколько отличаются от общепринятых. Это связано с особенностью Mathcad как вычислительной системы. Рассмотрим основные действия по изменению формул.

Операторы могут быть унарными (действующими на один операнд, как, например, оператор транспонирования матрицы или смены знака числа), так и бинарными (например, + или /, действующими на два операнда). При вставке нового оператора в документ Mathcad определяет, сколько операндов ему требуется. Если в точке вставки оператора один или оба операнда отсутствуют, Mathcad автоматически помещает рядом с оператором один или два местозаполнителя.

Чтобы произвести вставку оператора в формулу, нужно поместить линию ввода на часть формулы, которая должна стать первым операндом, а затем ввести оператор, нажав кнопку на панели инструментов или сочетание клавиш. Для того, чтобы вставить оператор не после, а перед частью формулы, выделенной линиями ввода, нажмите перед его вводом клавишу <Ins>, которая передвинет вертикальную линию ввода вперед. Это важно, в частности, для вставки оператора отрицания. На рис. 3.5.2-18 показаны примеры вставки оператора в различные части формулы.

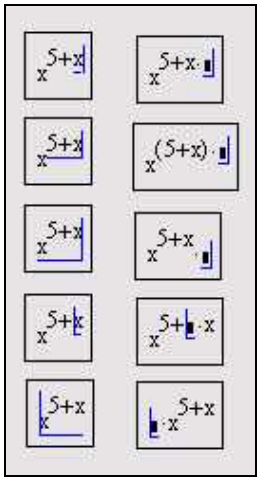


Рис. 3.5.2-18. Вставка оператора в разные части формулы

Некоторые операторы Mathcad вставит в правильное место независимо от положения линий ввода. Таков, например, оператор численного вывода =, который по смыслу выдает значение всей формулы в виде числа.

Многие операции редактирования связаны с необходимостью удаления, перемещения или копирования не одного символа, а фрагмента формулы, всей формулы или даже нескольких объектов документа. Перед выполнением перечисленных действий редактируемую часть документа следует выделить.

Чтобы выделить часть формулы в некоторой математической области с помощью клавиш, достаточно установить курсор перед (или после) выделяемой областью и, используя клавиши стрелки при удерживаемой клавише «Shift», выделить требуемый участок формулы. При этом выделенная часть формулы станет черной (рис. 3.5.2-19).

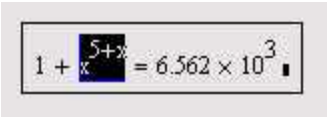


Рис. 3.5.2-19. Выделение части формулы

Часть формулы можно выделить и с помощью мыши. Для этого указатель мыши помещают на вертикальную линию ввода до (или после) выделяемой области, и, нажав и удерживая левую кнопку мыши, выделяют требуемый участок формулы.

Чтобы удалить часть формулы, ее надо выделить и нажать клавишу <Del>». Кроме того, можно удалить часть формулы, помещая ее перед вертикальной линией ввода и нажимая клавишу <BackSpace>.

Имеется еще один способ удаления части формулы: выделите ее нужную часть, затем нажмите комбинацию клавиш <Ctrl>+<X>, тем самым вырезая и помещая ее в буфер обмена. Этот способ удобен в случае, если требуется использовать фрагмент формулы в дальнейшем.

Команды редактирования (вставить, удалить, копировать) применимы не только к отдельно взятой формуле, но и группе объектов документа Mathcad (текста, функций,

результатов выполнения и т.д.), но предварительно их следует выделить. Выделение объекта или группы объектов осуществляется нажатием кнопки мыши в свободном месте окна и растяжением пунктирного прямоугольника таким образом, чтобы он охватил нужные объекты. Затем кнопку мыши следует отпустить.

Вместо команд главного меню часто используют команды контекстно-зависимого меню - это меню, которое открывается системой в результате щелчка правой кнопкой мыши по соответствующему объекту, расположенному в окне Mathcad. Это меню (рис. 3.5.2-20) содержит набор команд, которые в данный момент можно выполнить по отношению к конкретному объекту.

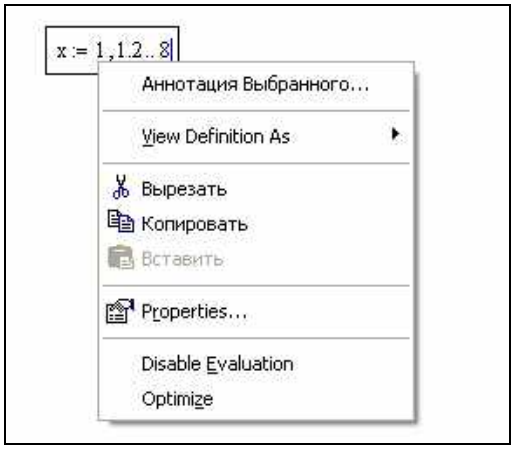


Рис. 3.5.2-20

Документы Mathcad могут содержать текстовые объекты, а также разного рода комментарии и примечания. Для того чтобы ввести текст непосредственно в рабочую область документа Mathcad, достаточно перед началом ввода текста нажать клавишу <">. В результате, в месте расположения курсора ввода появится область с характерным выделением, обозначающая, что ее содержимое не будет восприниматься процессором Mathcad в качестве формул, а станет простым текстовым блоком (рис. 3.5.2-21). Редактировать атрибуты текста в пределах блоков можно стандартными для текстовых редакторов средствами панели Форматирование.

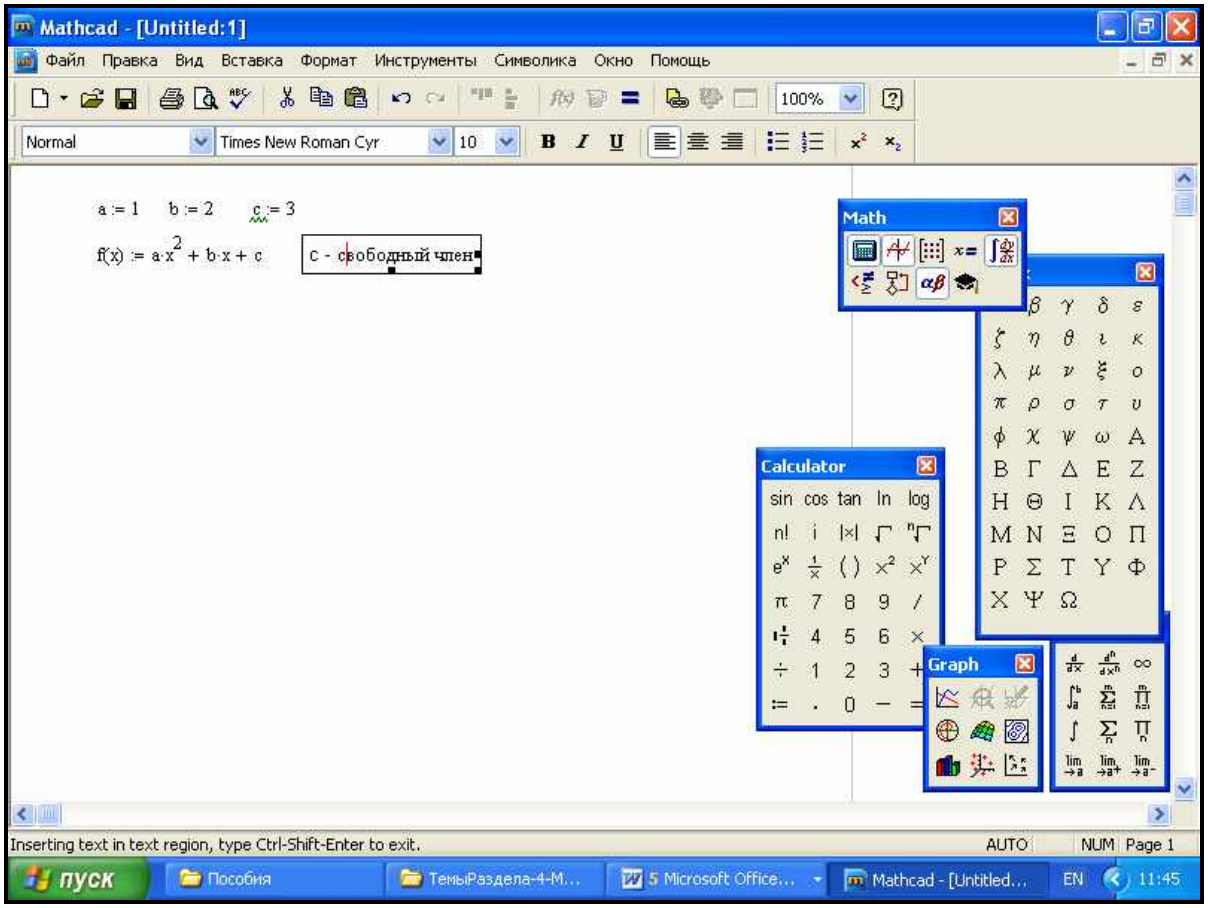


Рис. 3.5.2-21. Комментарий к фрагменту формулы

3.5.2.3. Построение графиков и визуализация результатов вычислений

Графический редактор Mathcad обеспечивает построение различных видов графиков: в декартовой и полярной системе координат, график поверхности, график уровней и т.п., которые можно разбить на две большие группы: двумерные (от одной переменной) и трехмерные (от двух переменных).

Для построения графиков в подменю График (позиция Вставка) имеются шаблоны типов графиков. Большинство параметров, необходимых для построения графиков, задается автоматически. Вид окна системы с подменю вставки шаблонов показан на рис. 3.5.2-22.

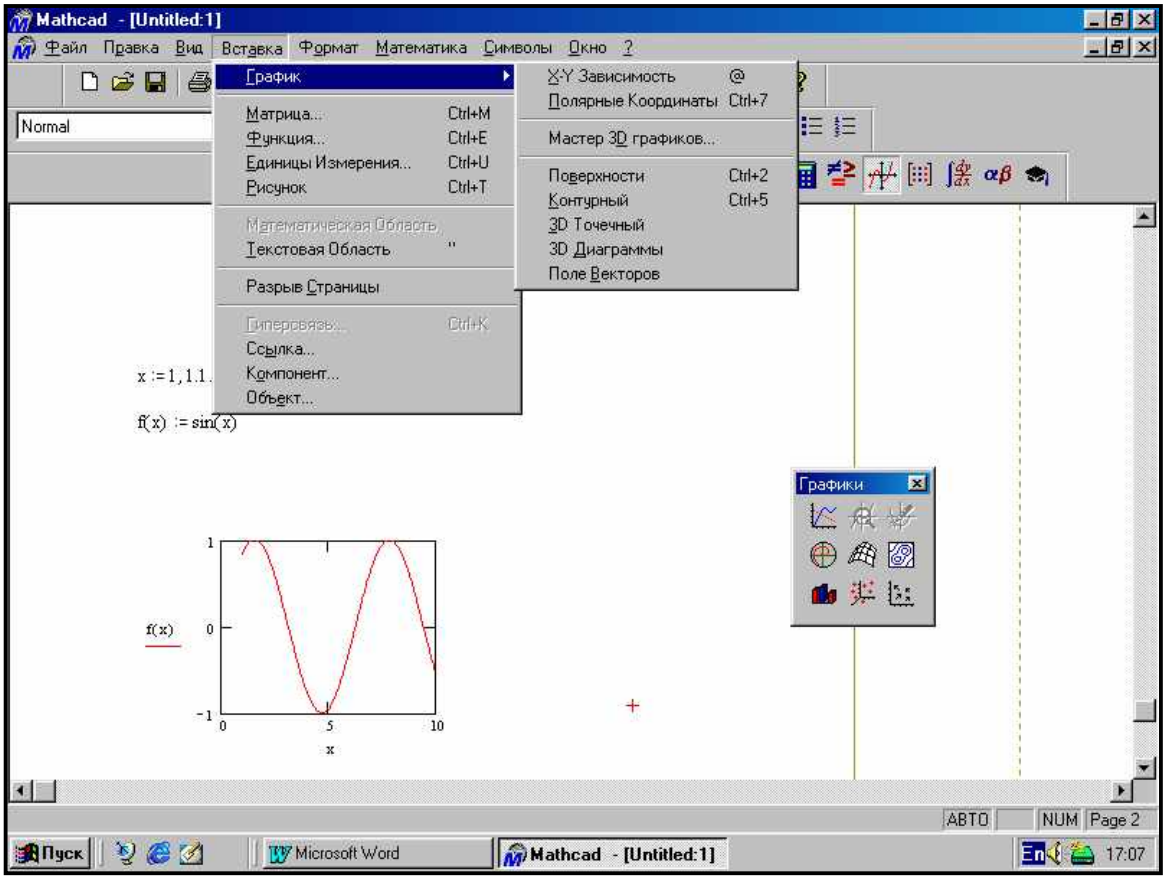


Рис. 3.5.2-22. Вид подменю вставки шаблона графики

Другим способом выбора шаблона типа графика является использование панели Графики, вызываемой щелчком по соответствующей кнопке панели Математика.

Перед построением графика функция, для которой он строится, должна быть описана, а ее аргументу (или аргументам) задан диапазон изменения. Если диапазон изменения аргумента не задан, то по умолчанию график будет построен в диапазоне от -10 до 10 с шагом 1. Для задания нужного диапазона и шага изменения аргумента в Mathcad используется дискретная переменная.

Рассмотрим **«Построение графиков одной переменной».**

Процедура построения графика функции от одной переменной заключается в следующем. Курсор устанавливается в то место, где должен располагаться график, затем на панели График щелчком по кнопке X-Y Plot (двумерный график) вызывается его формат, где в местозаполнитель по оси абсцисс вводится имя аргумента, а в местозаполнитель по оси ординат имя функции (рис. 3.5.2-23).

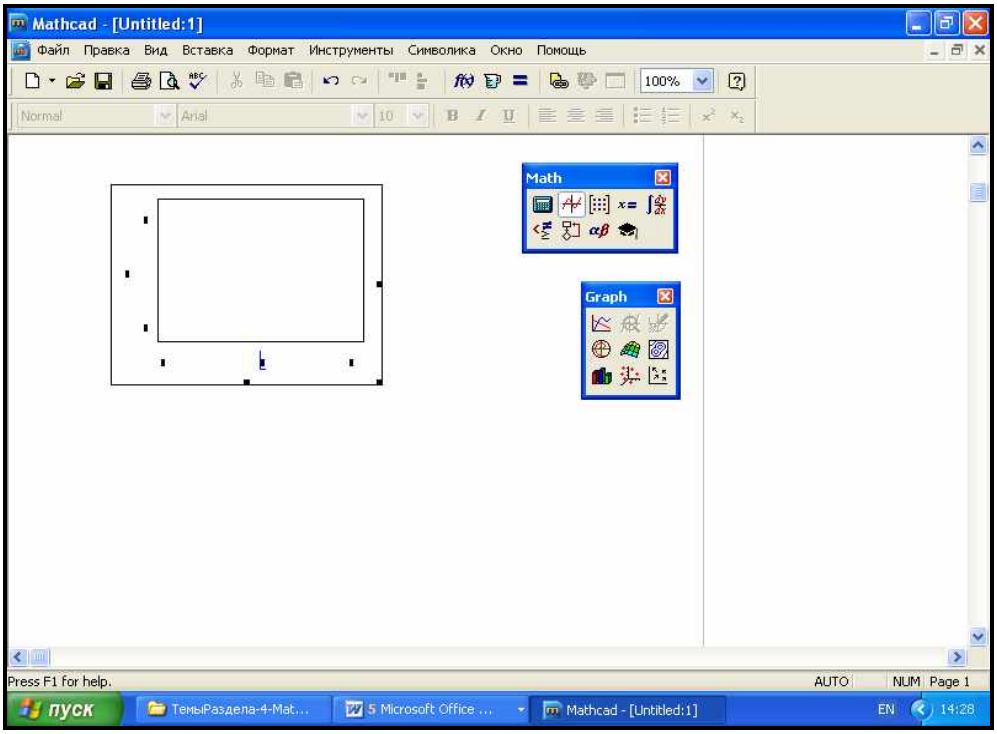


Рис. 3.5.2-23. Создание декартового графика при помощи панели Graph

Чтобы график функции появился, необходимо щелкнуть мышью вне графика.

Самый простой и наглядный способ получения графика в декартовой системе координат - это формирование двух векторов данных, которые будут отложены вдоль осей X и Y (рис. 3.5.2-24). В данном примере для задания индекса после имени вектора нужно нажать клавишу < [> (открывающая квадратная скобка). При этом курсор ввода устанавливается в месте ввода нижнего индекса.

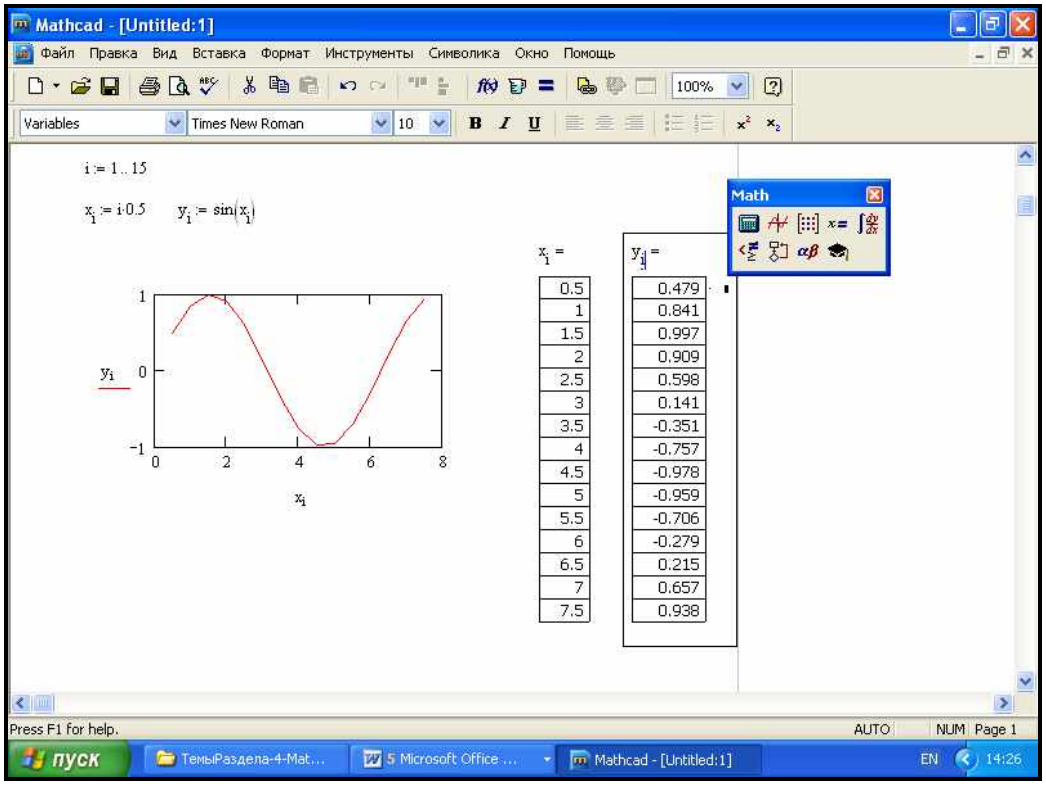


Рис. 3.5.2-24. График двух векторов

При построении в одном шаблоне двух и более графиков, имена функций в местозаполнителе вводятся через запятую. Если функции имеют разные аргументы, то имена аргументов также вводятся через запятую в последовательности, соответствующей последовательности введенных имен функций. На рис. 3.5.2-25 приведен пример построения графиков трех функций, аргументом которых является дискретная переменная x, изменяющаяся в диапазоне от 1 до 10 с шагом 0,2.

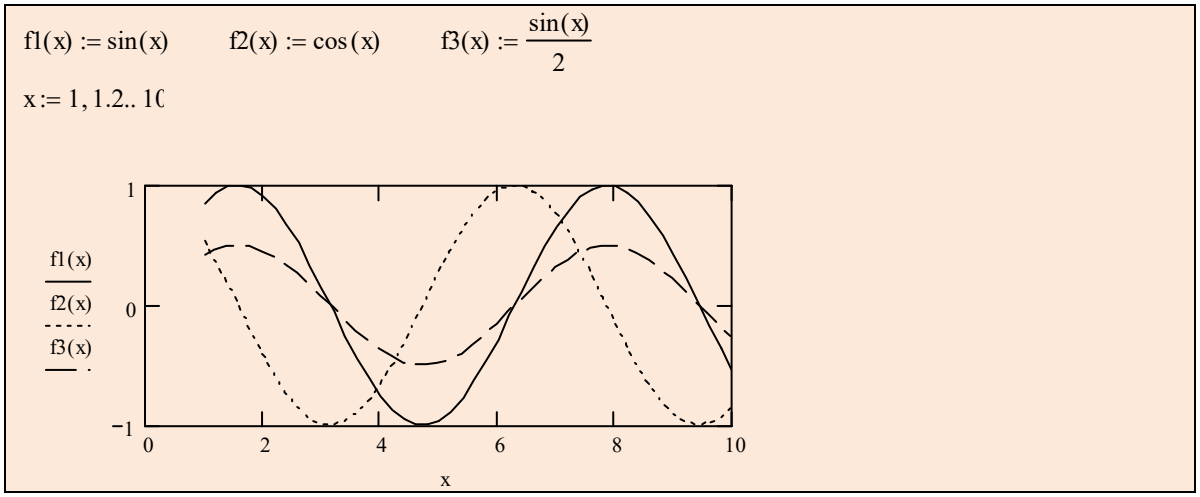


Рис. 3.5.2-25. Пример построения графика трех функций

Формат графика можно изменить с использованием установок, производимых в различных вкладках окна форматирования (рис. 3.5.2-26) (вызов окна – два щелчка мыши в области графика).

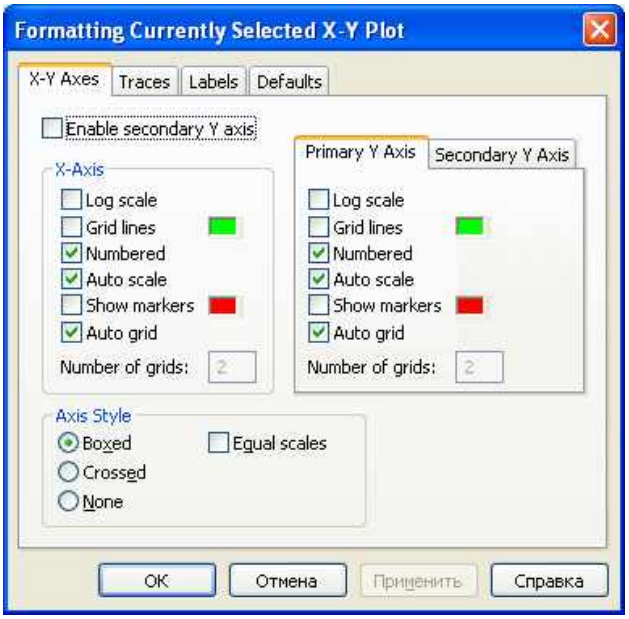


Рис. 3.5.2-26. Окно форматирования графиков

С использованием вкладки X-Y Axes (Оси X-Y) можно провести форматирование осей координат (нанести сетку, оси пересечения и т.п.). Установки вкладки Traces (Следы) позволяют провести форматирование самих графиков (вид линии, цвет, толщина линии и т.п.). Вкладка Labels (Метки) предназначена для внесения заголовков в область графика, а с помощью вкладки Defaults (Умолчание) можно вернуться к виду графика принятому по умолчанию.

Чтобы удалить график, нужно щелкнуть в его пределах мышью и выбрать в верхнем меню Правка пункт Вырезать или Удалить.

Рассмотрим построение **«Трехмерных изображений».**

Чтобы создать трехмерный график, требуется нажать кнопку с изображением любого из типов трехмерных графиков на панели инструментов Графики. В результате появится пустая область графика с тремя осями (рис. 3.5.2-27) и единственным местозаполнителем в нижнем левом углу. В этот местозаполнитель следует ввести, например имя z (имя функции от двух переменных - z(x,y)).

Для построения трехмерных графиков в современных версиях Mathcad появилась достаточно простая процедура, которая состоит в следующем: объявить двумерную функцию с использованием оператора присваивания (:=), а затем, установив курсор в место ввода графика, щелкнуть на панели График по кнопке Трехмерный график. На месте курсора появится шаблон трехмерного графика, в единственное место шаблона которого вводится имя функции. График появится после щелчка вне области графика. Эта процедура была применена при построении левого графика (рис. 3.5.2-28).

График поверхности может быть построен и с использованием массива численных значений функции (на рис. 3.5.2-29 справа). При этом, с помощью дискретных переменных должны быть введены значения обоих аргументов заданной функции, затем введен массив, элементами которого являются значения функции, и только после этого введен шаблон графика, в котором задается имя функции. График будет построен после щелчка мышью вне области шаблона.

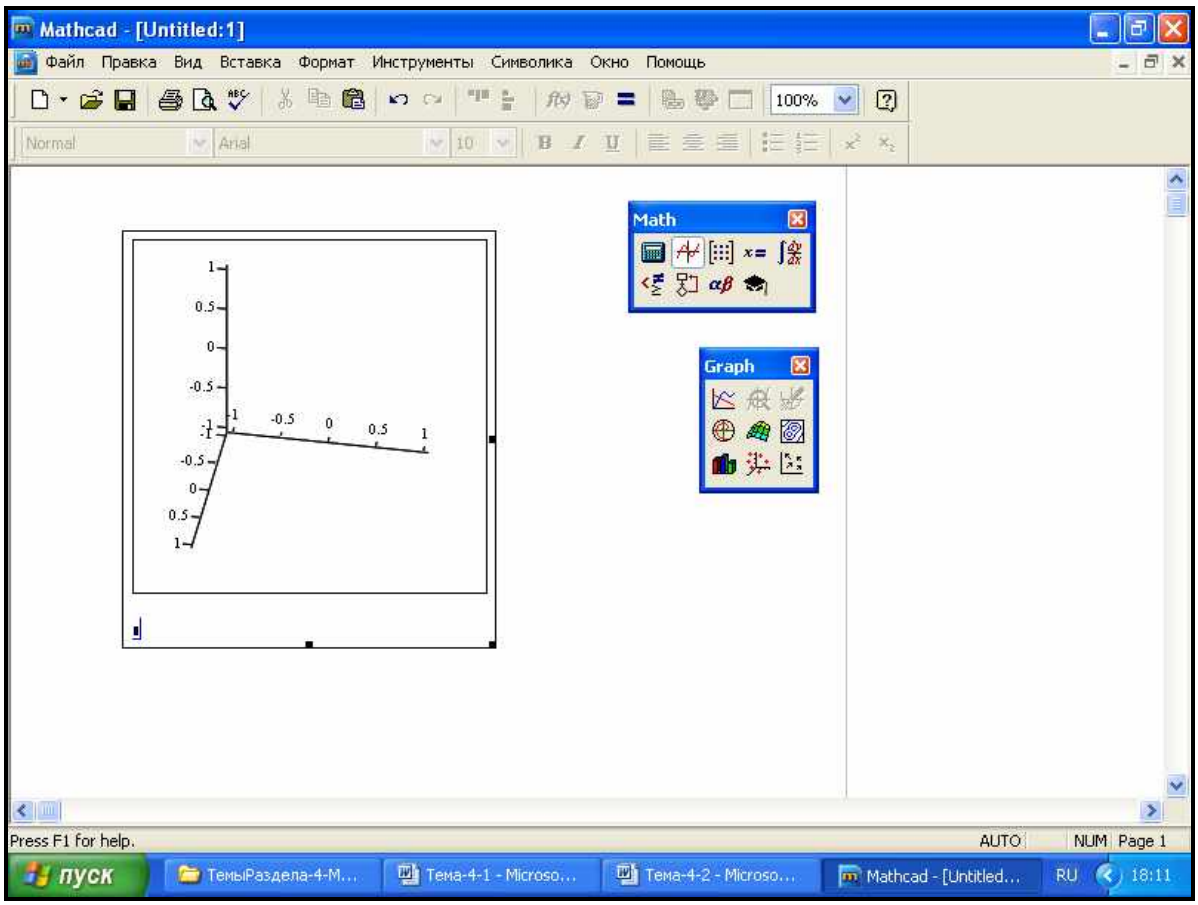


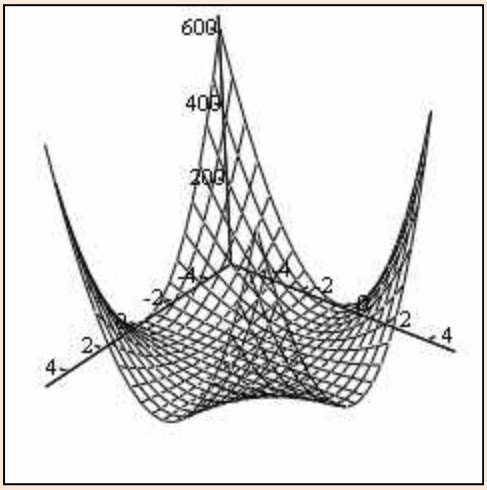
Рис. 3.5.2-27. Создание шаблона трехмерного графика

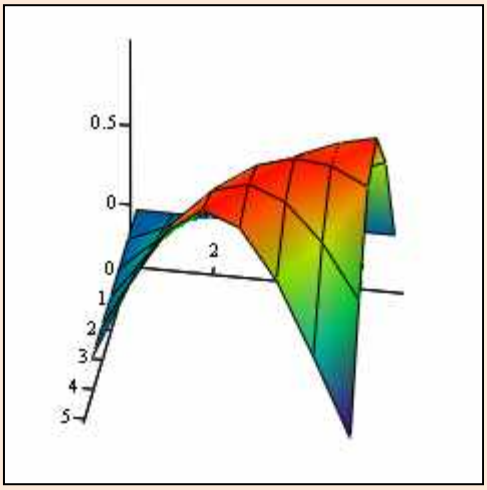
$n := 5 \qquad m := 7$

$i := 0..n \qquad j := 0..m$

$x_i := 0.5 \cdot i \qquad y_j := 0.2 \cdot j$

$Z_{i,j} := \sin(x_i \cdot y_j) \qquad f(x,y) := (x^2 \cdot y^2)$

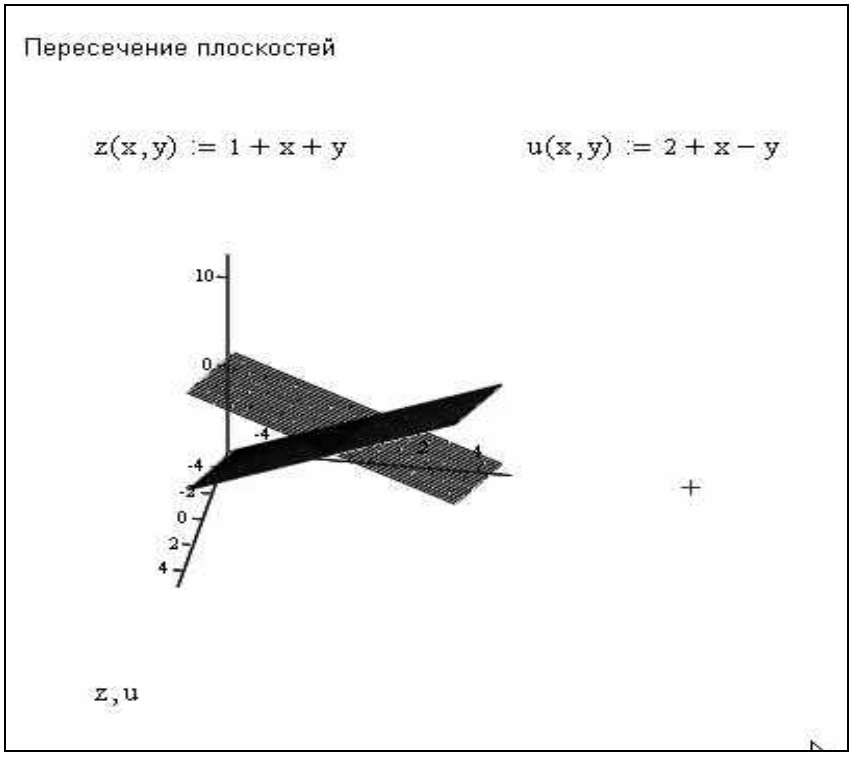




f Z

Рис. 3.5.2-28. Построение трехмерных графиков

В одном шаблоне трехмерного графика можно построить график не только одной, но и нескольких двумерных функций. В качестве примера на рис. 3.5.2-29 приведены графики пересечения двух плоскостей (две функции) и построение тора (три функции). В этом случае в местозаполнитель через запятую требуется ввести имена обеих функций.



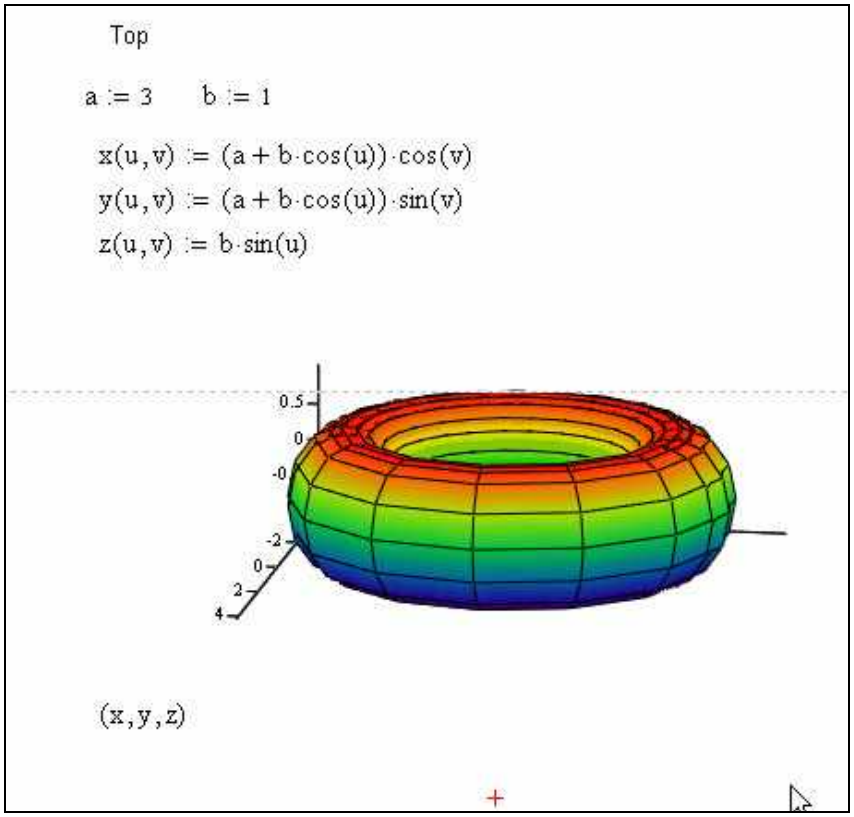


Рис. 3.5.2-29. Одновременное построение графиков нескольких двумерных функций

Для форматирования трехмерных графиков используются настройки окна, вызываемого двумя щелчками в области графика (рис. 3.5.2-30).

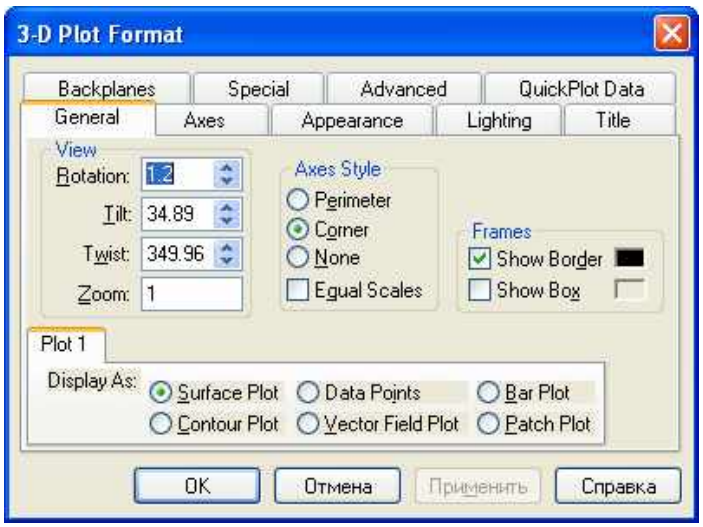


Рис. 3.5.2-30

В окне форматирования на вкладке General (Общие) Display As (Показать как) имеется 6 переключателей, позволяющих выбрать тип графика:

- Surface plot – график поверхности;
- Contour plot – график линий уровня;
- Data points – на графике представлены только расчетные точки;
- Vector Field plot – график векторного воля;
- Bar plot – график трехмерной диаграммы;
- Patch plot – площадка расчетных значений.

Чтобы выбрать тот или иной тип графика, нужно установить соответствующий переключатель и щелкнуть по кнопке Применить.

Например, если для графика функции $\mathbf{f(x,y) = x^2 + y^2}$, приведенного на рис. 3.5.2-31, выбрать тип Bar plot (график трехмерной диаграммы), то график приобретет вид как на рис. 3.5.2-32.

Трехмерный график можно вращать и масштабировать. Вращение графика выполняется наведением на него указателя мыши при нажатой левой кнопке. Масштабирование выполняется аналогично, но при нажатой дополнительно клавише <Ctrl>. Анимация графика (медленное вращение) выполняется также аналогично, но при нажатой кнопке <Shift>.

При построении графика в полярной системе координат (рис. 3.5.2-32), каждая точка задается углом W и модулем радиус-вектора R(W) . График функции обычно строится в виде линий, которую описывает конец радиус-вектора при изменении угла W в определенных пределах (чаще всего от 0 до 2π). После вывода шаблона надо ввести W в шаблон снизу, а R(W) – справа.

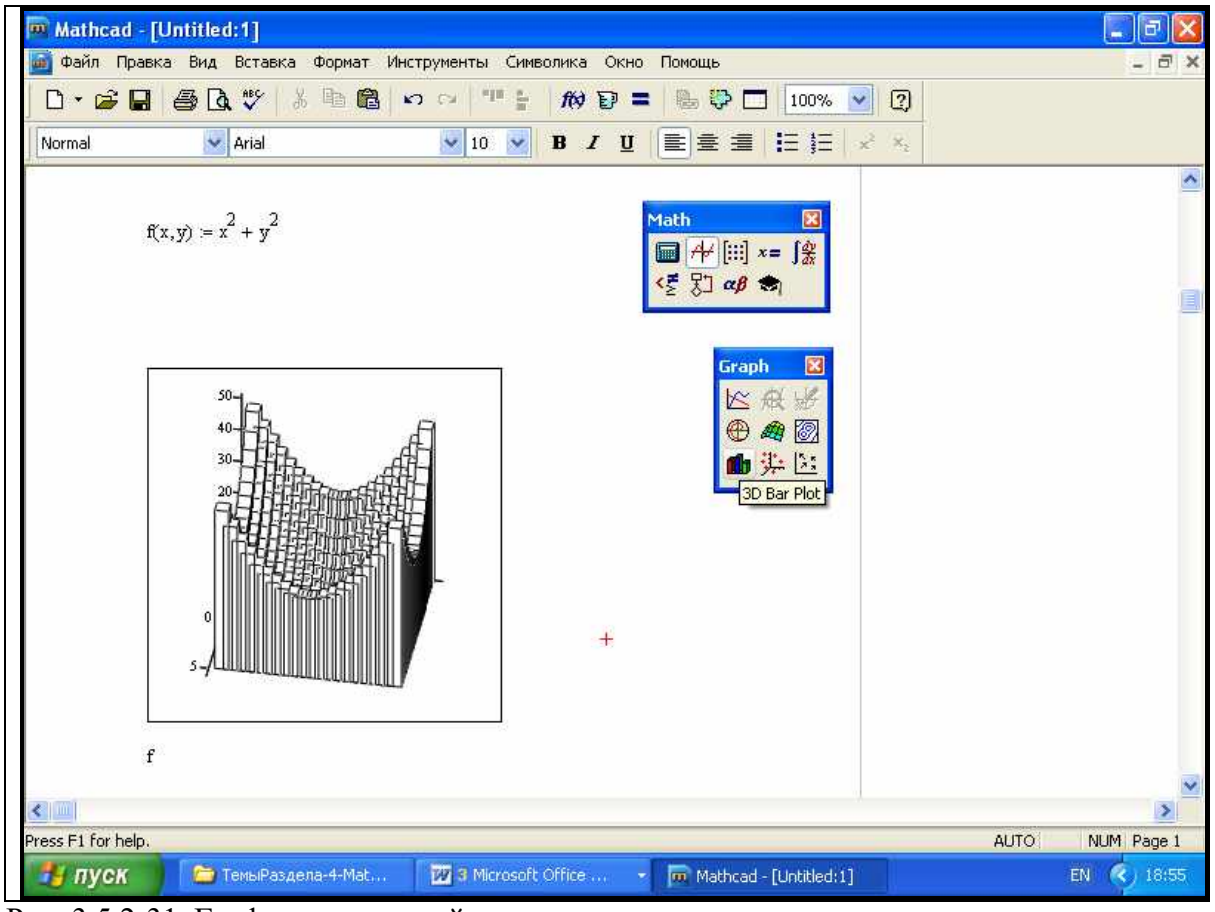
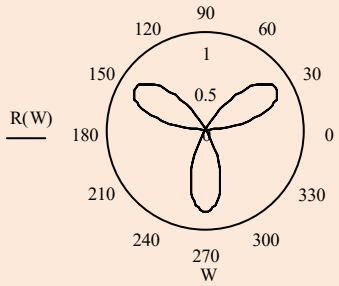


Рис. 3.5.2-31. График трехмерной диаграммы

$W := 0, 0.01 \cdot \pi .. 2 \cdot \pi$

$R(W) := \sin(3 \cdot W)$ $Rmin := 0$ $Rmax := 1.2$



$n := 20$

$i := 0 .. n$

$\phi_i := i \cdot \frac{\pi}{n}$

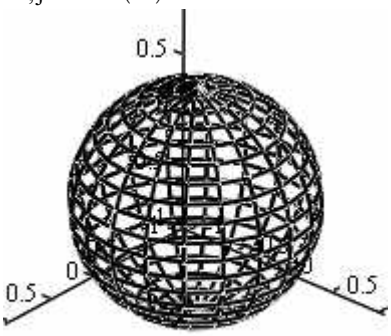
$j := 0 .. n$

$\Theta_j := j \cdot 2 \cdot \frac{\pi}{n}$

$X_{i,j} := \sin(\phi_i) \cdot \cos(\Theta_j)$

$Y_{i,j} := \sin(\phi_i) \cdot \sin(\Theta_j)$

$Z_{i,j} := \cos(\phi_i)$



(X, Y, Z)

Рис. 3.5.2-32. Построение графиков в полярной системе координат

Еще один широко распространенный тип графиков для представления поверхностей – контурный график представления поверхностей с помощью линий равного уровня (рис. 3.5.2-33). Тип графика задается с помощью соответствующей кнопки панели График.

$z(x,y) := -\sin(x \cdot y)$ $x := 0..20$ $y := 0..20$

$$M_{(x,y)} := f\left[\frac{(x-10)}{3}, \frac{(y-10)}{3}\right]$$

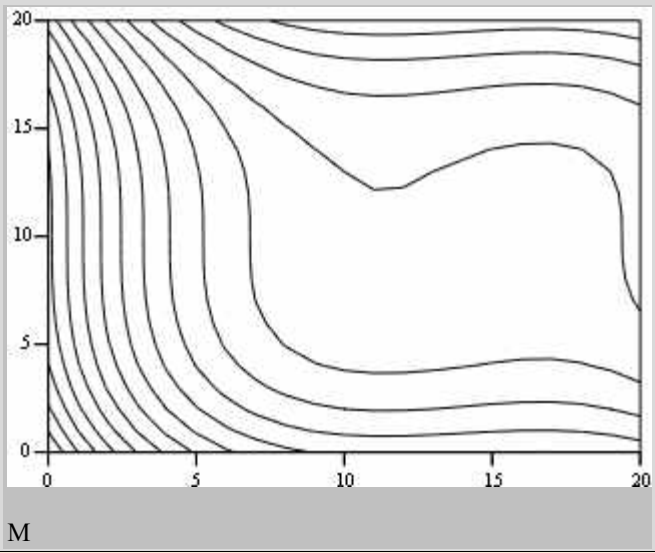


Рис. 3.5.2-33. Построение контурного графика

3.5.2.4. Символьные алгебраические преобразования

Символьный процессор **Mathcad** позволяет решать многие задачи математики аналитически, без применения численных методов и, следовательно, без погрешностей вычислений. В символьных вычислениях допускается использование большинства встроенных функций **MathCad**, конечно, за исключением тех, которые реализуют численные методы. Символьный процессор умеет выполнять основные алгебраические преобразования, такие, как **упрощение выражений**, **разложение их на множители, символьное суммирование и перемножение.**

Символьные вычисления в **MathCad** можно осуществлять в двух различных вариантах:

- с помощью команд меню;
- с помощью оператора символьного вывода —>.

Первый способ более удобен, когда требуется быстро получить какой-либо аналитический результат для однократного использования, не сохраняя сам ход вычислений. Второй способ более нагляден, т. к. позволяет записывать выражения в традиционной математической форме и сохранять символьные вычисления в документах **MathCad**. Кроме того, аналитические преобразования, проводимые через меню, касаются только одного, выделенного в данный момент, выражения. Именно поэтому в дальнейшем будем рассматривать примеры символьного преобразования с использованием панели **Symbolic**.

На панели **Symbolic** (Символика) находятся кнопки, соответствующие специфическим командам символьных преобразований (рис. 3.5.2-34). Например, таким, как разложение выражения на множители, приведение подобных слагаемых и другим операциям, которые в **Mathcad** нельзя проводить численно, и для которых, соответственно, не предусмотрены встроенные функции.

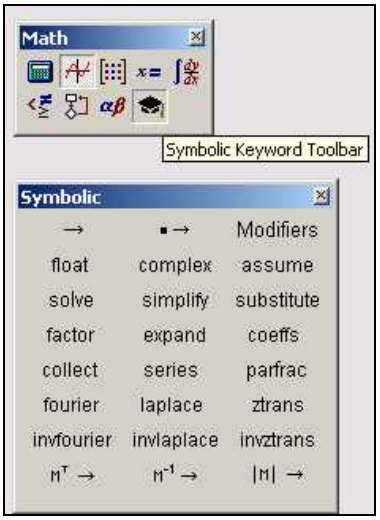


Рис. 3.5.2-34. Панель Simbolic (Символика)

Рассмотрим символьное вычисление на примере **разложения на сомножители** выражения **cos(4x)** с помощью оператора -> и команды **Expand** (Разложить), отраженной в виде кнопки на панели **Symbolic** (Символика). Для этого требуется выполнить последовательность следующих действий:

1. Введите выражение **cos (4x)**.
2. Нажмите кнопку **Expand** (Разложить) на панели **Symbolic** (Символика).
3. Введите в местозаполнитель после появившегося ключевого слова **expand** (рис. 3.5.2-35) имя переменной **x,** либо нажмите клавишу <**Del**>, чтобы просто удалить местозаполнитель.
4. Введите оператор символьного вывода —>.
5. Нажмите клавишу <**Enter**> либо просто щелкните мышью за пределами выражения.

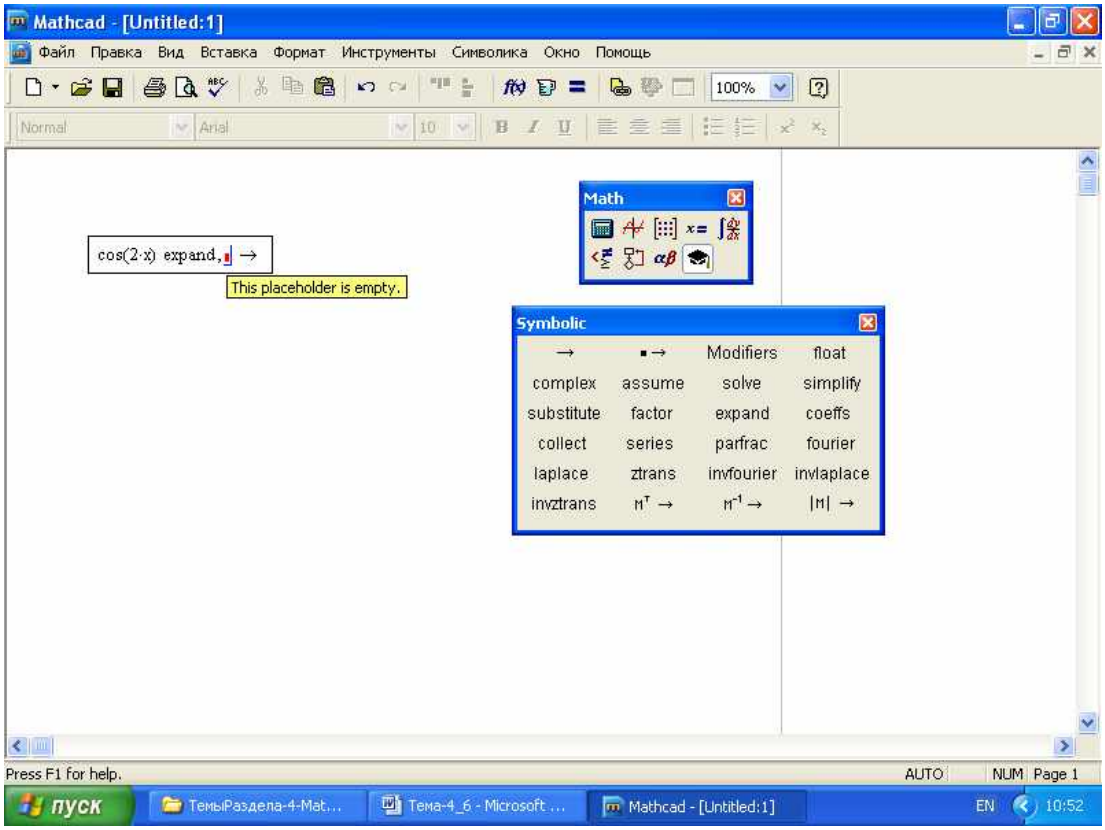


Рис. 3.5.2-35. Символьное разложение выражения

Отметим, что можно действовать и в другом порядке: сначала ввести ключевое слово, а уже затем впечатать, в появившиеся местозаполнители, выражение и переменную (рис. 3.5.2-36).

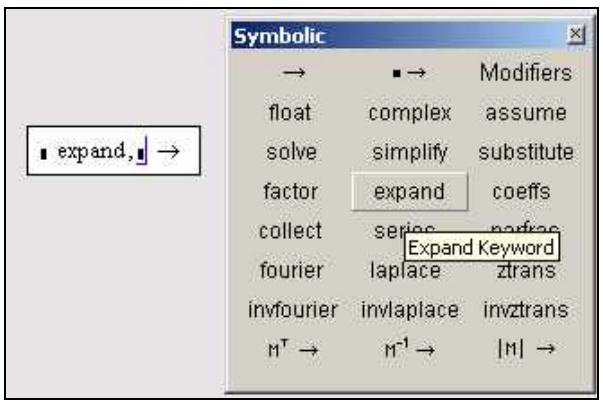


Рис. 3.5.2-36. Результат первоначального ввода ключевого слова

Необходимо помнить, что не всякое выражение поддается аналитическим преобразованиям. Если это так (например, если задача вовсе не имеет аналитического решения, или она оказывается слишком сложной для символьного процессора **MathCad**), то в качестве результата выводится само выражение, например рис. 5.4.1-4.

$$\cos(2x)\ \text{expand},x\ \rightarrow 2\cdot\cos(x)^2 - 1$$
$$\cos(x)\ \text{expand},x\ \rightarrow \cos(x)$$

Рис. 3.5.2-36

Очень часто применяемая операция - **упрощение выражений**, по смыслу эта операции противоположна операции разложения. Символьный процессор **MathCad** стремится так преобразовать выражение, чтобы оно приобрело более простую форму. При этом используются различные арифметические формулы, приведение подобных слагаемых, тригонометрические тождества, пересчет обратных функций и др.

Для упрощения выражения при помощи оператора символьного вывода используйте ключевое слово **simplify** (рис. 3.5.2-37).

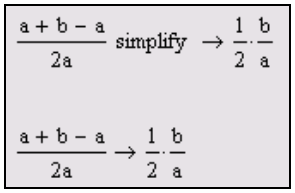


Рис. 3.5.2-37. Упрощения математического выражения

Следует помнить, что если некоторым переменным, входящим в выражение, ранее были присвоены некоторые значения, то они будут подставлены в него при выполнении символьного вывода (рис. 3.5.2-38).

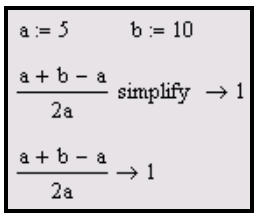


Рис. 3.5.2-38. Упрощение выражения с подстановкой значения переменных

Упрощение выражений, содержащих числа, производится по-разному, в зависимости от наличия в числах десятичной точки. Если она есть, то выполняется непосредственное вычисление выражения (рис. 3.5.2-39).

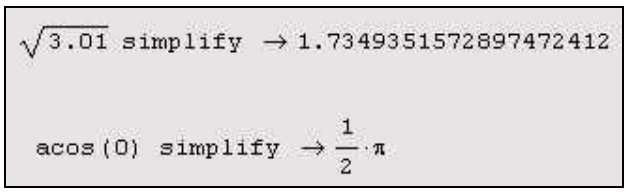


Рис. 3.5.2-39. Упрощение выражения с числами.

С помощью ключевого слова **factor** (Разложить на множители), расположенного напанели **Symbolic**, **MathCad** позволяет производить **разложение выражений на простые множители** (рис. 3.5.2-40). Эта операция позволяет разложить полиномы на произведение более простых полиномов, а целые числа — на простые сомножители.

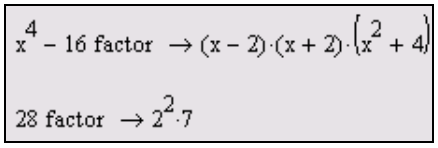


Рис 3.5.2-40. Примеры разложения на множители.

Чтобы привести **подобные слагаемые** с помощью оператора символьного вывода, нужно произвести следующую последовательность действий:

1. Введите выражение.
2. Нажмите кнопку **Collect** на панели **Symbolic** (Символика).
3. Введите в местозаполнитель после вставленного ключевого слова **collect** имя переменной, относительно которой требуется привести подобные слагаемые (на рис. 3.5.2-40 в первой строке - это переменная **x**, во второй — **y**).
4. Введите оператор символьного вывода —>.
5. Нажмите клавишу <**Enter**>.

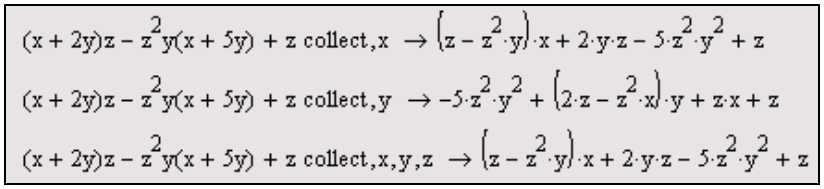


Рис. 3.5.2-40. Приведение подобных слагаемых по разным переменным

Отметим, что после ключевого слова **collect** допускается задание нескольких переменных через запятую. В этом случае приведение подобных слагаемых выполняется последовательно по всем переменным.

Если выражение является **полиномом** относительно некоторой переменной x, заданным не в обычном виде **a0+a1x+a2x²+...**, а как произведение других, более простых полиномов, то символьным процессором **MathCad** легко **определяются полиномиальные коэффициенты a0,a1,a2....** Коэффициенты сами могут быть функциями (подчас довольно сложными) других переменных.

Чтобы вычислить полиномиальные коэффициенты с помощью оператора символьного вывода, следует выполнить следующие действия:

1. Введите выражение.
2. Нажмите кнопку **Coeffs** на панели **Symbolic** (Символика).
3. Введите в местозаполнитель после вставленного ключевого слова **coeffs** аргумент полинома.
4. Введите оператор символьного вывода —>.
5. Нажмите клавишу <**Enter**>.

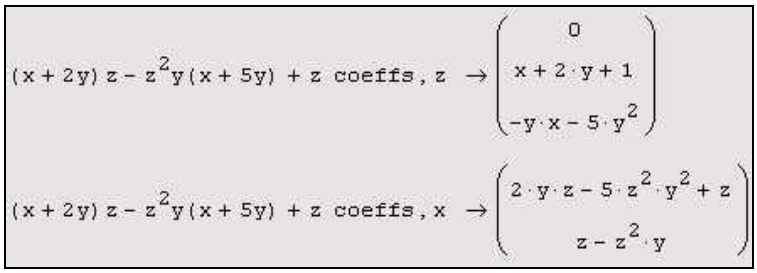


Рис. 3.5.2-41. Вычисление полиномиальных коэффициентов для простой переменной и выражения

Чтобы разложить сложную дробь на более простые дроби, следует выполнить команду **Convert to Partial Fractions** (Разложить на элементарные дроби), для этого на панели **Symbolic** (Символика) достаточно указать ключевое слово **parfrac** (рис. 3.5.2-42), имя переменной указывается после ключевого слова **parfrac**.

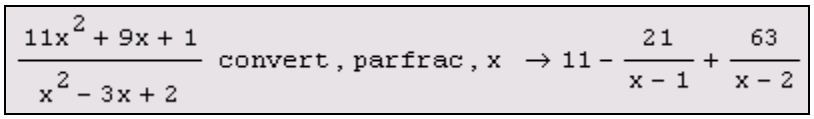


Рис. 3.5.2-42. Разложение на элементарные дроби.

3.5.2.5. Аналитическое решение задач математического анализа

Для аналитического решения математических задач в **Mathcad** используется панель **Calculus** (Вычисления) (рис. 3.5.2-43).

Рис. 3.5.2-43. Панель **Calculus** (**Вычисления**).

Чтобы вычислить аналитически **конечную или бесконечную сумму** или **произведение**:

1. Введите выражение, используя панель **Calculus** (**Вычисления**) для вставки соответствующих символов суммирования или произведения. При необходимости введите в качестве предела ряда символ бесконечности, находящейся на той же панели.
2. Введите оператор символьного вывода —>.

На рис. 3.5.2-44. приведены примеры численного и символьного вычисления рядов и произведений.

$$\sum_{i=0}^{10} 2^i = 2.047 \times 10^3 \qquad \sum_{i=0}^{10} 2^i \to 2047 \qquad \sum_{i=0}^{\infty} a^i \to \frac{-1}{a-1}$$

$$\prod_{n=1}^{4} \frac{1}{n^3+1} = 3.053 \times 10^{-5} \qquad \prod_{n=1}^{4} \frac{1}{n^3+1} \to \frac{1}{32760} \qquad \prod_{n=1}^{\infty} \frac{1}{n^3+1} \to 0 \qquad \prod_{n=1}^{\infty} \sqrt{n} \to \infty$$

Рис. 3.5.2-44. Символьные и числовые расчеты сумм и произведений

Очень удобная возможность символьных вычислений - это **операция подстановки значения переменной в выражение**. Для осуществления этой операции в совокупности с оператором символьного вывода используйте ключевое слово **substitute**, которое вставляется в документ одноименной кнопкой на панели **Symbolic** (Символика). После ключевого слова **substitute** необходимо ввести в местозаполнители логическое выражение, показывающее, какую именно переменную, какой формулой следует заменить.

$$\sin(k \cdot x^2 + b \cdot x) \text{ substitute}, k = a \cdot x^2 \to \sin(a \cdot x^4 + b \cdot x)$$

Рис. 3.5.2-45. Подстановка значения переменной

С помощью символьного процессора можно **рассчитать численное значение выражения (действительное или комплексное).** Иногда такой путь представляется более удобным, чем применение численного процессора (т. е. знака обычного равенства). Чтобы рассчитать значение некоторого выражения, с панели **Symbolic** (Символика) следует ввести ключевое слово **float** или **complex**. Ключевое слово **float** применяется вместе со значением точности вывода результата с плавающей точкой (Рис. 3.5.2-46). С помощью слова **complex** можно преобразовывать выражения как в символьном виде, так и с учетом численных значений, если они были ранее присвоены переменным (Рис. 3.5.2-47).

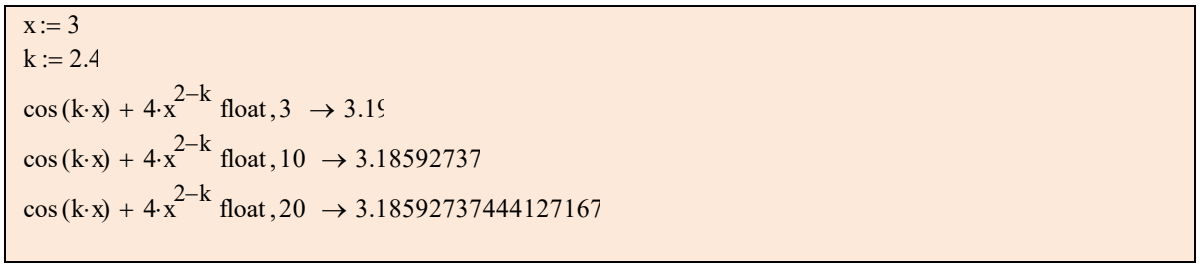


Рис. 3.5.2-46. Вычисление выражения с плавающей точкой.

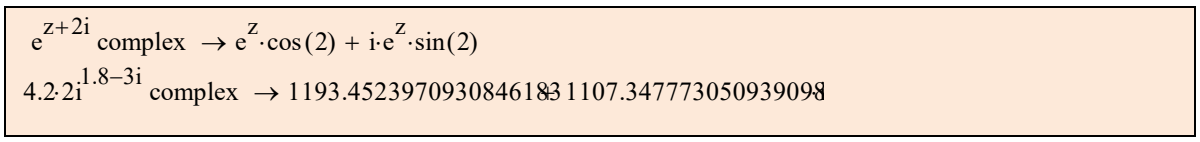


Рис. 3.5.2-47. Комплексные преобразования выражений.

Наиболее ярким проявлением возможностей символьного процессора в **MathCad** являются аналитические **вычисления пределов, производных, интегралов и разложений в ряд**, а также **решение алгебраических уравнений**. Все эти операции легко выполняются с использованием операторов, расположенных на панели **Symbolics** (Символика).

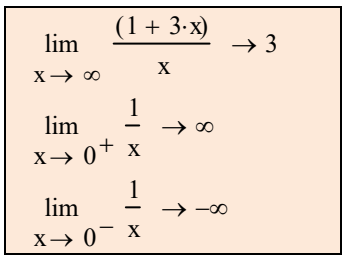


Рис. 3.5.2-48. Операторы символьного вычисления пределов.

Для того, чтобы аналитически найти производную функции **f(x)** в **MathCad,** необходимо**:**

1. Задать функцию **f (x)**.
2. Ввести оператор дифференцирования нажатием кнопки **Derivative** (Производная) на панели **Calculus** (Вычисления).
3. В появившихся местозаполнителях оператора дифференцирования (рис. 3.5.2-49) ввести функцию, зависящую от аргумента **x**, т. е. **f(x)**, и имя самого аргумента **x**.
4. Ввести оператор символьного вычисления ->.

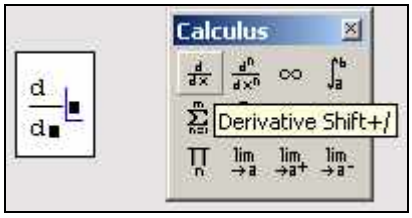


Рис. 3.5.2-49. Оператор дифференцирования

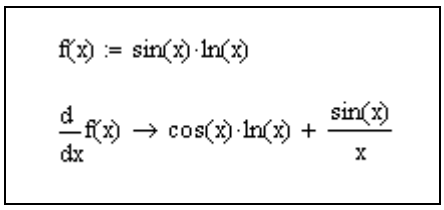


Рис. 3.5.2-50. Пример аналитического дифференцирования.

Для того, чтобы рассчитать производную в точке, необходимо предварительно задать значение аргумента в этой точке. Результатом дифференцирования в этом случае будет число - значение производной в этой точке (рис. 3.5.2-51).

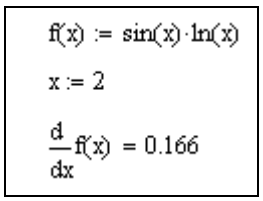


Рис. 3.5.2-51. Аналитическое дифференцирование функции в точке.

Для того чтобы продифференцировать функцию, вовсе не обязательно предварительно присваивать ей какое-либо имя, можно определить функцию непосредственно в операторе дифференцирования (рис. 3.5.2-52).

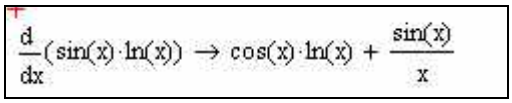


Рис. 3.5.2-52

Интегрирование в **MathCad** реализовано в виде вычислительного оператора. Допускается вычислять интегралы от скалярных функций в пределах интегрирования, которые также должны быть скалярными. Несмотря на то, что пределы интегрирования обязаны быть действительными, подынтегральная функция может иметь и комплексные значения, поэтому и значение интеграла может быть комплексным.

Интегрирование, как и дифференцирование, и множество других математических действий, устроено в **MathCad** по принципу "как пишется, так и вводится". Чтобы вычислить определенный интеграл, следует напечатать его обычную математическую форму в документе. Делается это с помощью панели **Calculus** (Вычисления) нажатием кнопки со значком интеграла. Появится символ интеграла с несколькими местозаполнителями (рис. 3.5.2-53), в которые нужно ввести нижний и верхний интервалы интегрирования, подынтегральную функцию и переменную интегрирования.

Если пределы интегрирования имеют размерность, то она должна быть одной и той же для обоих пределов.

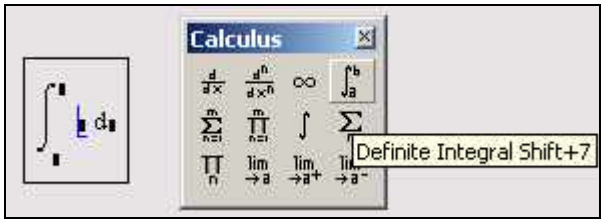


Рис. 3.5.2-53. Оператор интегрирования

Чтобы получить результат интегрирования, следует ввести знак равенства или символьного равенства. В первом случае интегрирование будет проведено численным методом, во втором - будет найдено точное значение интеграла с помощью символьного процессора **MathCad** (рис. 3.5.2-54). Конечно, символьное интегрирование возможно только для сравнительно небольшого круга несложных подынтегральных функций.

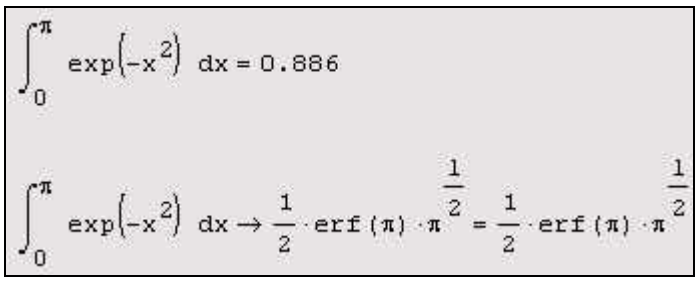


Рис. 3.5.2-54. Численное и символьное вычисление определенного интеграла

Можно вычислять интегралы с одним или обоими бесконечными пределами (рис. 3.5.2-55). Для этого на месте соответствующего предела введите символ бесконечности воспользовавшись, например, той же самой панелью **Calculus** (Вычисления).

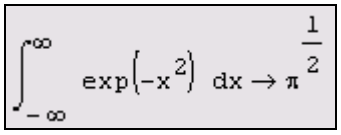


Рис. 3.5.2-55. Вычисление интеграла с бесконечными пределами.

Подынтегральная функция может зависеть от любого количества переменных. Именно для того, чтобы указать, по какой переменной **MathCad** следует вычислять интеграл, и нужно вводить ее имя в соответствующий местозаполнитель.

Для того, чтобы **аналитически проинтегрировать** некоторую функцию, следует ввести с панели **Calculus** (Вычисления) символ **неопределенного интеграла** (рис. 3.5.2-56).

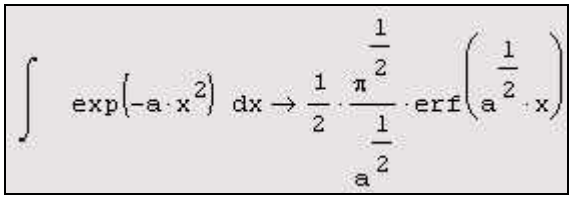


Рис. 3.5.2-56. Аналитическое вычисление неопределенного интеграла.

Если интеграл расходится (равен бесконечности), то вычислительный процессор **MathCad** может выдать сообщение об ошибке, выделив при этом оператор интегрирования, как обычно, красным цветом. Чаще всего ошибка будет иметь тип **"Found a number with a magnitude greater than 10^{307}"** (Найдено число, превышающее значение 10^{307}) или **"Can't converge to a solution"** (Не сходится к решению). На рис. 3.5.2-57 показан случай

невозможности численного расчета интеграла, тем не менее символьный процессор справляется с этим интегралом, совершенно правильно находя его бесконечное значение.

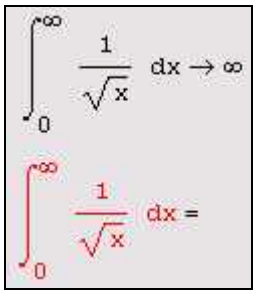


Рис. 3.5.2-57. Вычисление расходящегося интеграла.

Кратным называется интеграл функции многих переменных, берущийся по нескольким переменным. Для того чтобы вычислить кратный интеграл:

1. Введите, как обычно, оператор интегрирования.
2. В соответствующих местозаполнителях введите имя первой переменной интегрирования и пределы интегрирования по этой переменной.
3. На месте ввода подынтегральной функции введите еще один оператор интегрирования (рис. 3.5.2-58).
4. Точно так же введите вторую переменную, пределы интегрирования и подынтегральную функцию (если интеграл двукратный) или следующий оператор интегрирования (если более чем двукратный) и т. д., пока выражение с многократным интегралом не будет введено окончательно.

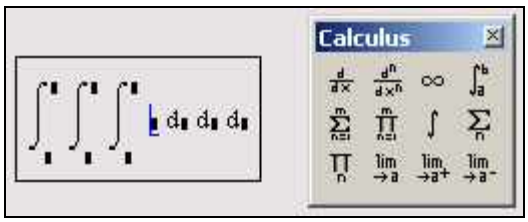


Рис. 3.5.2-58. Ввод нескольких операторов интегрирования для расчета кратного интеграла

Пример символьного и численного расчета двукратного интеграла в бесконечных пределах приведен на рис. 3.5.2-59. Обратите внимание, что символьный процессор "угадывает" точное значение интеграла, а вычислительный - определяет его приближенно и выдает в виде числа **3.142**.

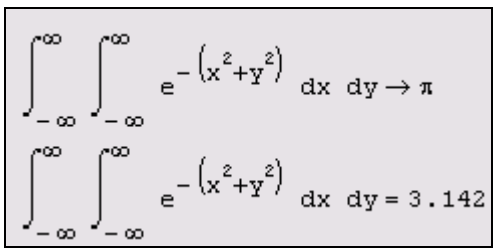


Рис. 3.5.2-59. Символьное и численное вычисления кратного интеграла.

3.5.2.6. Оформление и печать документа Mathcad

Помимо того, что **Mathcad** является мощным математическим инструментом, в нем еще предусмотрены возможности оформления внешнего вида расчетов. Основными элементами оформления документов в **Mathcad**, которые допускается применять как собственно для расчетов, так и в чисто декоративных целях, являются (рис. 3.5.2-60, сверху вниз):

- текстовые области;
- математические области;
- графики, или графические области;
- компоненты других приложений.

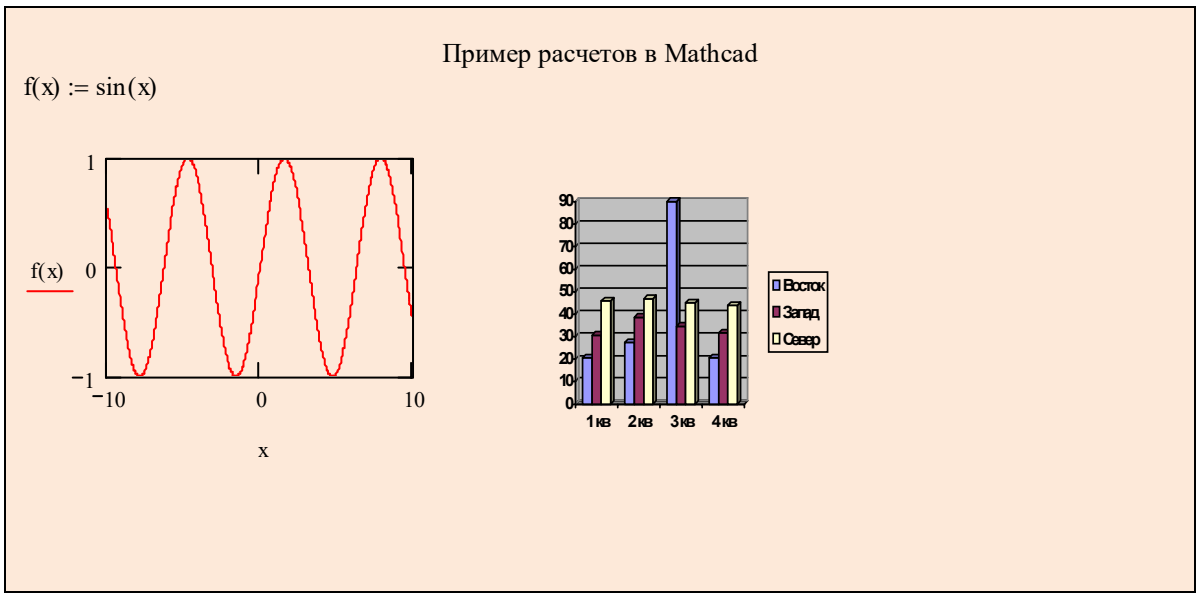


Рис. 3.5.2-60. Основные элементы оформления документа

Для вставки того или иного элемента нужно поместить курсор в нужное место и осуществить вставку либо с помощью меню **Вставка**, либо с помощью соответствующего элемента панели инструментов, либо, как в случае ввода формулы или текста, просто начать вводить символы с клавиатуры.

Местоположение элемента оформления можно изменить, для чего:

1. Щелкните в области элемента мышью. Область станет выделенной.
2. Не нажимая кнопок, поместите указатель мыши на границу области, чтобы курсор сменил вид стрелки на форму руки.
3. Нажмите левую кнопку мыши и, удерживая ее, перетащите объект на новое место.

Чтобы **создать копию** области в другом месте документа, перетаскивание следует выполнять при нажатой клавише **<Ctrl>.**

Рассматривая рис. 3.5.2-61, где выделенным элементом является область графика, можно заметить, что справа и снизу на границе выделенной области расположены черные прямоугольники. С их помощью осуществляется **растягивание** или **сжатие** области элемента. Прямоугольник в нижнем правом углу предназначен для одновременного изменения размеров области по горизонтали и вертикали.

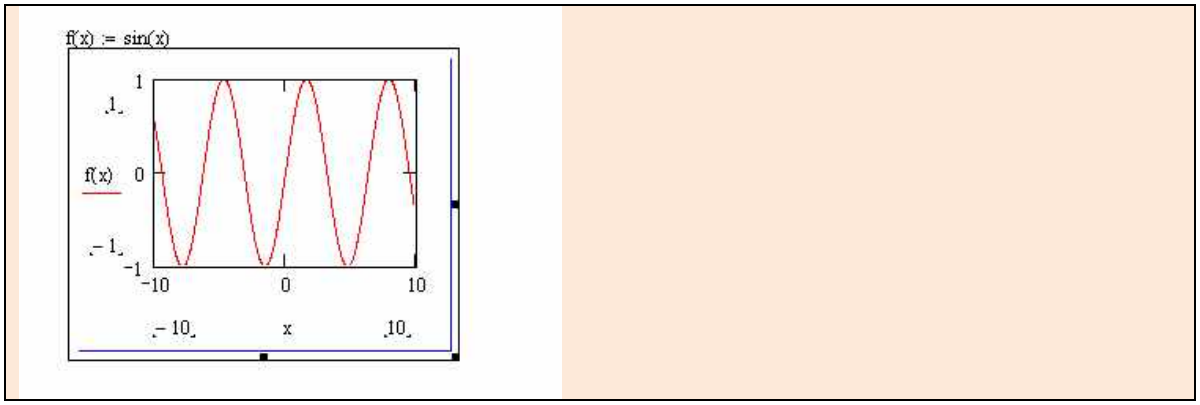


Рис. 3.5.2-61. Выделенный документ

В **Mathcad** области можно **выделять цветом**. Для этого достаточно выбрать в меню **Формат** пункт **Свойства** (Properties), а затем установить в появившемся окне флажок **Highlight Region** (Выделить цветом) и нажать **OK**. По умолчанию цвет выделения желтый. Чтобы установить нужный цвет, нужно щелкнуть по кнопке **Choose Color** (Выбрать цвет) и установить нужный цвет (рис. 3.5.2-62).

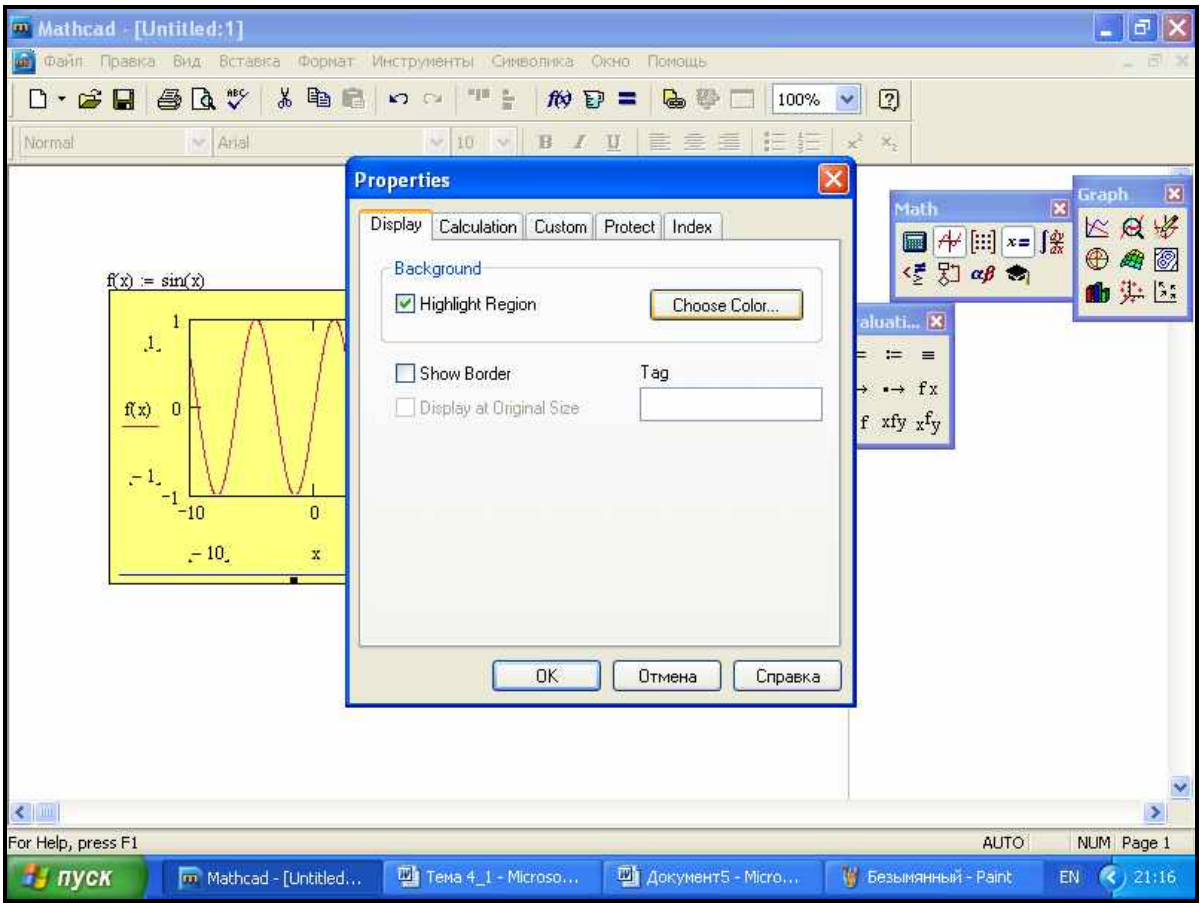


Рис. 3.5.2-62. Выделение области цветом

Выделить область можно не только цветом, но и **обрамлением**. Для этого в том же самом окне **Properties** следует установить флажок **Show Border** (Показать рамку). Обычно эта операция выполняется вместе с установкой цвета.

В начале раздела в нижней части на рис. 3.5.2-60 присутствует особый тип области, названный **компонентой другого приложения**. Эта область через буфер памяти была перенесена из другого приложения (в данном случае из текстового редактора **Word**) и

вставлена обычным образом в документ **Mathcad**. Следует отметить, что с внедренными областями в **Mathcad** можно обращаться точно также как и его собственными: копировать, удалять, изменять размеры и т.п. Однако на практике гораздо чаще приходится встречаться с противоположным действием – **экспортированием области документа Mathcad в другие приложения.**

Чтобы вставить определенную область документа **Mathcad** в другое приложение нужно:

1. С помощью мыши произвести выделение нужных элементов;
2. Из контекстного меню или щелчком по соответствующей кнопке панели выполнить команду **Копировать** (выделенная область помещается в буфер);
3. Перейти в окно документа куда экспортируется область и, установив курсор в место вставки, выполнить команду **Вставить**.

Чаще всего области документа **Mathcad** копируются в документы редактора **Word** и здесь есть следующая особенность. При редактировании документа **Word** вставленные объекты самопроизвольно перемещаются, а зачастую, и редактируются по своим правилам. Чтобы вставленный элемент зафиксировать в нужном месте, рекомендуется место вставки подготовить следующим образом:

1. В документе Word установить курсор в строке вставки объекта;
2. Создать таблицу, состоящую из одной ячейки (1 строка и 1 столбец);
3. При необходимости выделить таблицу нужным цветом (выбором в контекстном меню элемента **Границы и заливка**);
4. Поместив курсор в созданную таблицу, выполнить команду **Вставить**.

Границы таблицы можно удалить с использованием элемента панели форматирования (рис. 3.5.2-63).

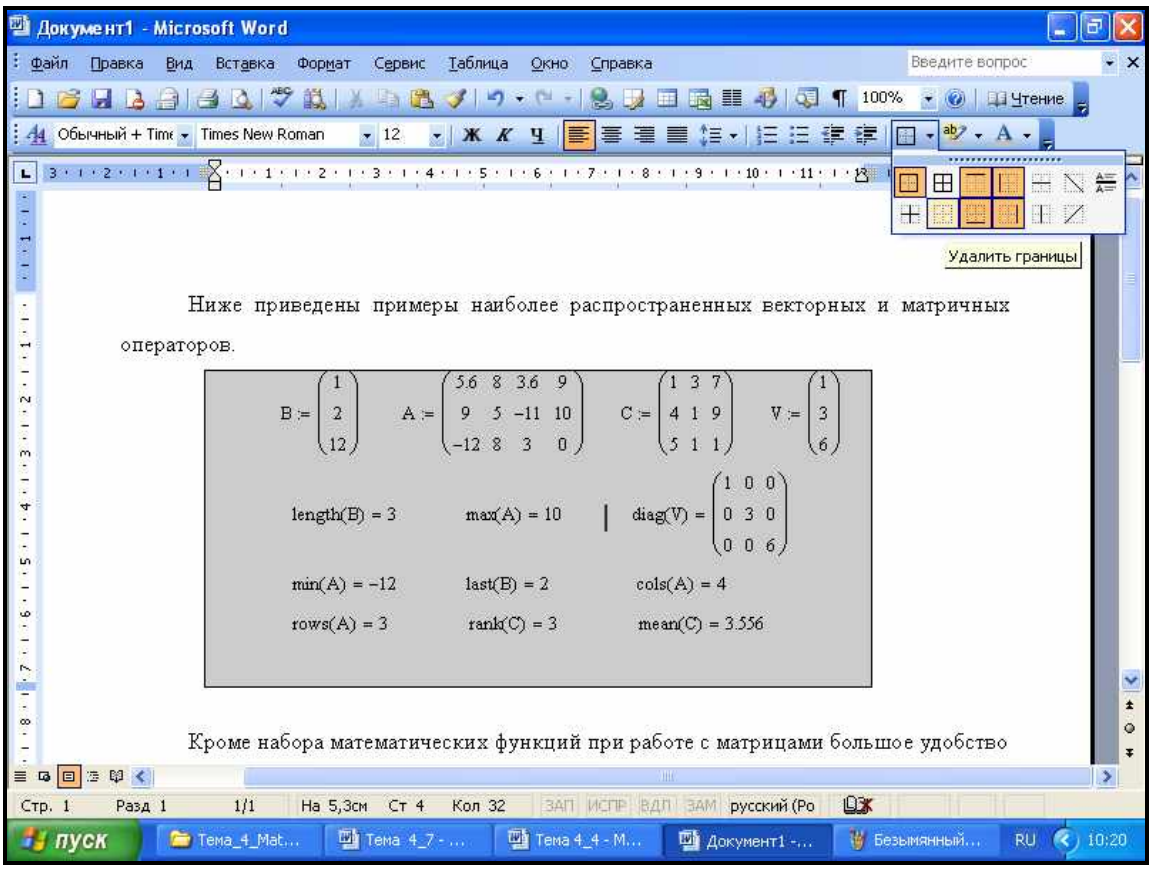


Рис. 3.5.2-63. Удаление границ таблицы

Текстовую область можно разместить в любом незанятом месте документа **Mathcad**. Однако, когда пользователь помещает курсор ввода в пустое место документа и просто начинает вводить символы, то **Mathcad** по умолчанию интерпретирует их как начало формулы. Чтобы до начала ввода указать, что требуется создать текстовый регион, достаточно перед вводом первого символа нажать клавишу <”> (двойная кавычка). При этом в месте ввода возникает текстовый регион, который имеет характерное выделение (рис. 3.5.2-64). Теперь можно вводить любые символы клавиатуры, которые будут восприниматься как текст.

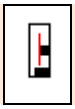


Рис. 3.5.2-64. Вновь созданный текстовый регион

Для форматирования текста и формул служит панель **Formatirung** (Форматирование). Форматирование текста в **Mathcad** во многом похоже на форматирование в большинстве текстовых редакторов. Оно заключается в управлении его двумя основными составляющими: форматом шрифта и форматом абзаца.

Шрифт выделенного текста можно поменять при помощи панели **Formatirung** (рис. 3.5.2-65, раскрыт список размеров шрифтов). Стилю шрифта на панели форматирования соответствуют кнопки:

- **Bold** (Полужирный);
- **Italie** (Наклонный);
- **Underlined** (Подчеркнутый),

а список доступных **типов шрифтов** раскрывается щелчком по стрелке второго элемента панели форматирования.

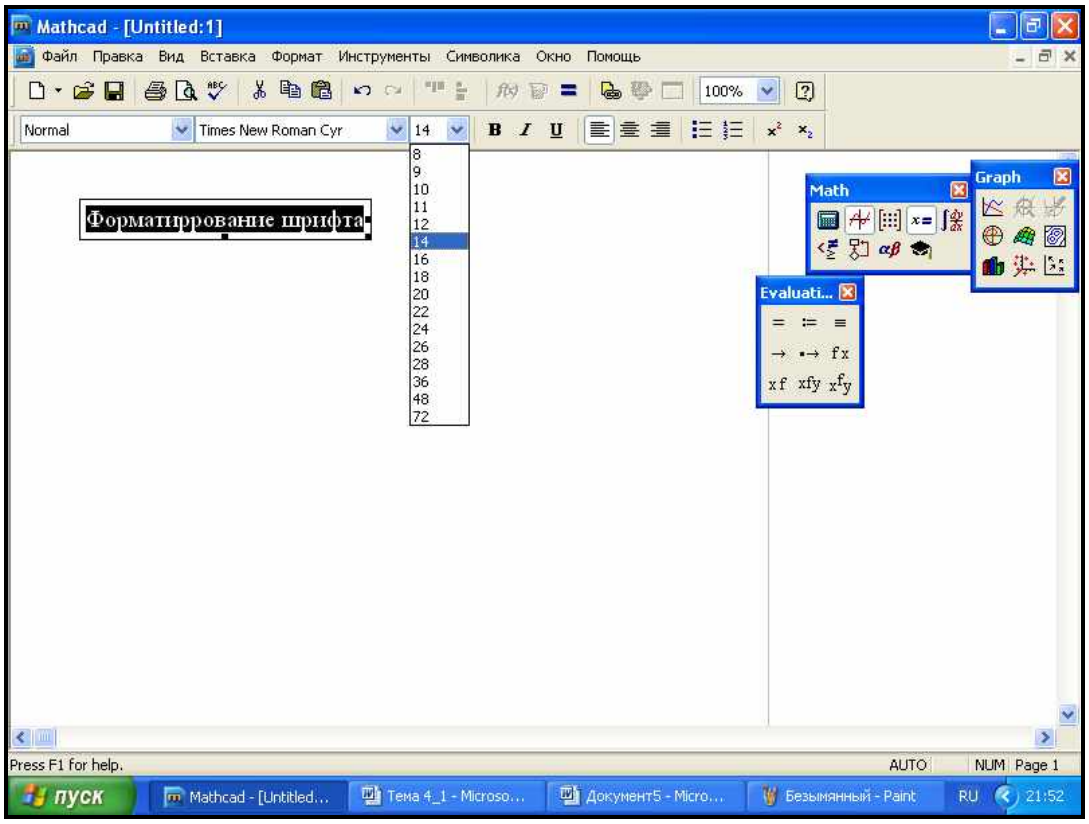


Рис. 3.5.2-65. Выбор размера шрифта

Параметры формата текста можно также установить и при помощи диалогового окна **Text Format** (рис. 3.5.2-66), вызываемого в верхнем меню **Формат**. Перечислим параметры шрифта и соответствующие элементы этой панели, которыми допускается управлять:

- **Font** (Шрифт);
- **Size** (Размер);
- **Font style** (Стиль шрифта).

Кроме того, в этом окне можно выбрать **цвет** (**Color**) и такие параметры как:

- **Strikeout** (Зачеркивание);
- **Underline** (Подчеркивание);
- **Superscript** (Верхний индекс);
- **Subseript** (Нижний индекс).

Для установки **параметров абзаца** применяются:

- **абзацный отступ** – три маркера на линейке в верхней части экрана (верхний левый маркер – левая граница первой строки; нижний левый маркер - все остальные строки; правый маркер – правая граница);
- **нумерованный и маркированный список** – две правые кнопки панели **Форматирование**;
- **выравнивание** – кнопки  панели **Форматирование**, соответственно, по левому краю, по центру и по правому краю.

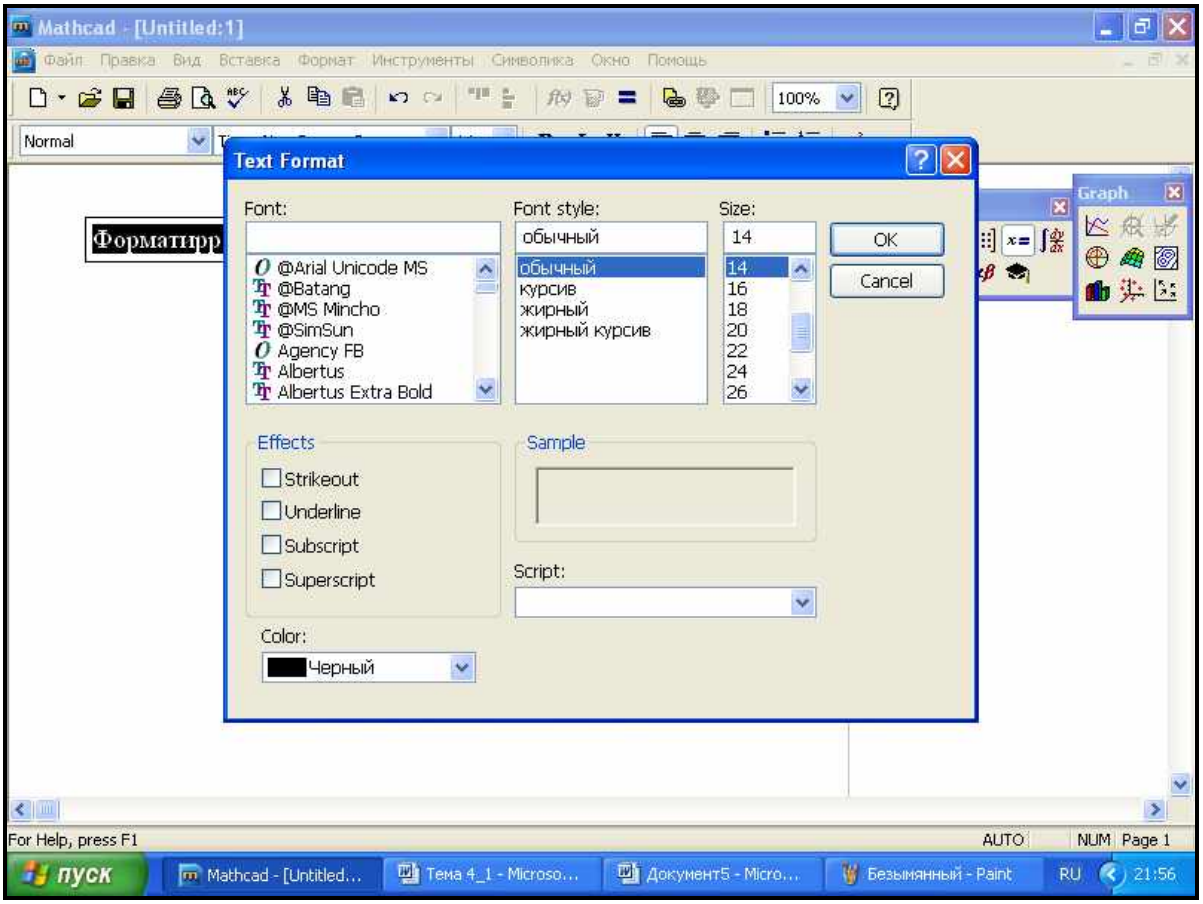


Рис. 3.5.2-66. Диалоговое окно **Text Format**

Для математических регионов (формул) можно применять все рассмотренные выше способы форматирования шрифта. Особенность форматирования формул заключается в том, что изменение шрифта, примененное к отдельному параметру в одной формуле, немедленно приводит к его изменению во всех формулах документа (там, где этот параметр присутствует). При этом надо помнить, что формулы содержат элементы, выполненные в нескольких математических стилях.

Чтобы распечатать экземпляр активного документа **Mathcad** на принтере, достаточно щелкнуть по кнопке с изображением принтера на панели инструментов **Стандартная**. Таким способом выводятся на печать небольшие документы или промежуточные результаты расчета. Следует ответить, что нажатие кнопки на панели инструментов приводит к мгновенной печати всего активного документа с текущими опциями печати и установками принтера.

Для более активного управления процессом печати служат следующие пункты меню **Файл**:

- **Установки страницы** – опции страницы вывода активного документа на печать (стандартный размер страницы, тип подачи бумаги, поля);
- **Просмотр** – предварительный просмотр на экране выводимого на печать активного документа;
- **Печать** – собственно печать активного документа с возможностью выбора принтера (если установлено несколько), смены установок принтера (качество печати, разрешение, количество печатных копий документа и диапазон печатаемых страниц).

Выбор любого из этих пунктов меню приводит к раскрытию одноименного диалогового окна.

3.5.3. Основы работы с математическим пакетом MatLab

3.5.3.1. Рабочая среда MatLab и простейшие вычисления в окне Command Window

3.5.3.1.1. Элементы основного окна MatLab

После запуска **MatLab** на экране дисплея появляется **основное окно** системы **MatLab** в стандартной конфигурации, показанное на рис.3.5.4-1. При этом система **MatLab** готова к проведению вычислений в **командном режиме (**в окне **Command Window).**

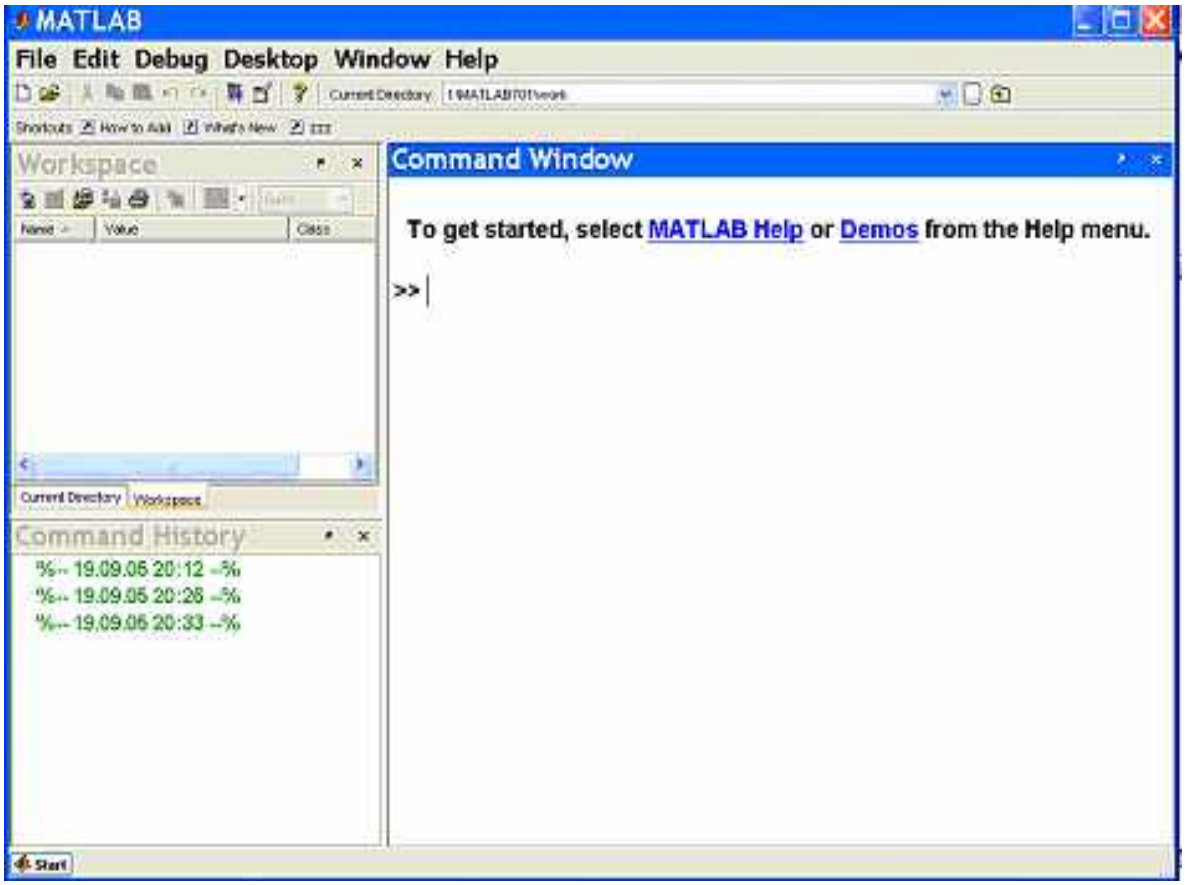


Рис. 3.5.3-1. **Основное окно** системы **MatLab** после запуска

Основное окно системы **MatLab** - это обычное окно приложений **Microsoft Windows**, поэтому его можно перемещать, изменять в размерах, открывать на весь экран.

В **основном окне** системы **MatLab** размещены:

- **панель главного меню,** которое является контекстно-зависимым;
- **главная панель инструментов**;
- **окно панели Command Window** (Окно Команд);
- **окно панели Command History** (Окно Истории Команд);
- **окно панели Workspace (**Окно Рабочей Области);
- **окно панели Current Directory** (Окно Текущего Каталога) – на рис. 3.5.3-1 показана вкладка для перехода в это окно;
- **кнопка Start**;
- **строка состояния**.

Две панели **Workspace** и **Current Directory** закрывают друг друга, и для активизации нужной панели необходимо щелкнуть по соответствующей вкладке. В стандартной конфигурации все три оконных панели, вписанные в основное окно **MatLab** (рис. 3.5.3-1), закреплены (поставлены на якорь), то есть они могут передвигаться вместе с основным окном и вместе с ним изменять свои размеры. Однако границы между этими окнами можно изменять, то есть каждое из них можно открепить (снять с якоря), и тогда оно может занимать автономную позицию на экране.

Панель главного меню системы **MatLab** расположена вверху основного окна и имеет шесть элементов, команды которых показаны на рис. 3.5.3-2, в предположении, что активным является окно панели **Command Window**:

File - содержит команды главного меню для работы с файлами команд и файлами рабочей области **MatLab**, а также команды установки параметров системы **MatLab**;

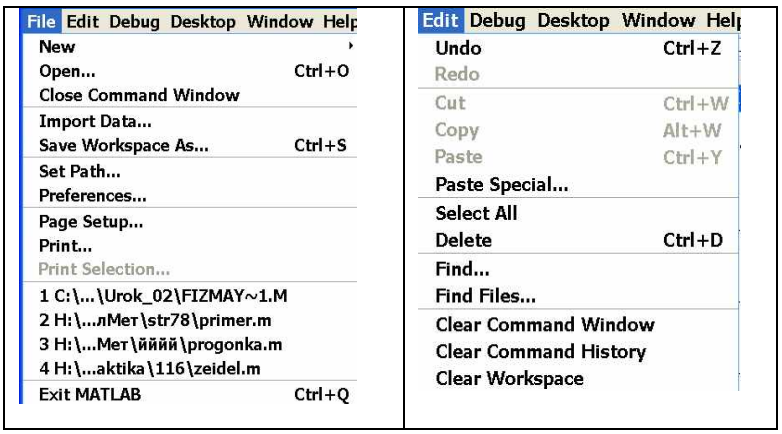
Edit - содержит команды главного меню для редактирования и очистки содержимого панелей основного окна;

Debug - содержит команды главного меню для работы с отладчиком **m-файлов**;

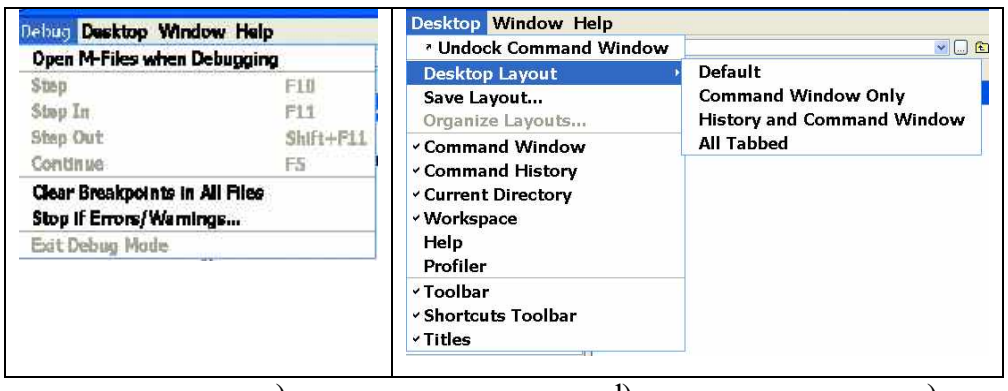
Desktop - содержит команды главного меню для управления панелями основного окна **MatLab**;

Windows - содержит команды главного меню для перевода в активное состояние различных окон **MatLab**;

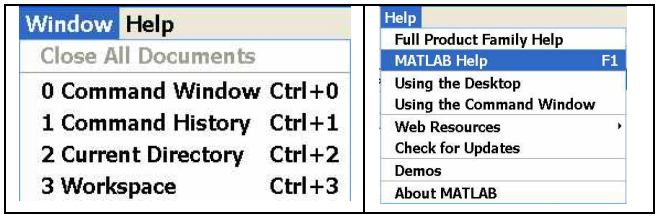
Help - содержит команды главного меню для доступа к элементам справочной системы **MatLab**.



a) b)



c) d) e)



f) g)

Рис. 3.5.3-2. Команды элементов главного меню системы **MatLab**

Элемент меню File, представленный на рис. 3.5.3-2a, содержит команды, позволяющие осуществлять операции с файлами, которые можно разделить по функциональным признакам на шесть частей.

С помощью команды **New** можно открыть окно редактора для создания и редактирования, программных **m**-файлов - окно **Editor**, графическое окно **Figure** для визуализации результатов вычислений**,** создать новую переменную**,** открыть окно динамической модели и графического интерфейса пользователя (**GUIDE** - Graphic User Interface Development Environment). Команда **Open** открывает стандартное диалоговое окно **MS Windows**, в котором можно выбрать необходимый файл. Команда **Close Command Window** закрывает командное окно.

Следующая группа команд позволяет импортировать данные (**Import Data...**) с диска в рабочую область системы **MatLab**, а также сохранять данные, расположенные в рабочей области, на диске (**Save Workspace as** ...).

Следующие две команды - **Set Path** и **Preferences** - открывают диалоговые окна, в которых можно настраивать список путей доступа к файлам и параметры работы системы **MatLab** соответственно. Список путей доступа используется при поиске необходимого файла во время выполнения команды. С помощью диалогового окна **Preferences** настраиваются параметры работы инструментальных средств системы **MatLab**: размер и цвет шрифта, цвет фона, формат вывода числовой информации и т.д.

Команды **Print Setup ..., Print... и Print Selection...** служат для открытия диалоговых окон, в которых производится настройка страницы для печати, выбор принтера и передача на печать содержимого текущего окна панели.

Следующая часть меню содержит до четырех последних имен файлов, с которыми производилась работа в системе **MatLab**.

Последняя команда - **Exit MATLAB** - осуществляет закрытие системы.

Элемент меню Edit, показанный на рис. 3.5.3-2b, содержит команды, которые позволяют: отменять (**Undo**) или возвращать (**Redo**) отмененное действие; обмениваться информацией с буфером обмена (**Cut** - Вырезать в буфер, **Copy** - Копировать в буфер, **Paste** - Вставить из буфера, **Paste Special** - Специальная вставка); полностью выделять содержимое текущего окна панели (**Select All**); удалять выделенный текст (**Delete**); осуществлять поиск фрагмента текста в текущем окне панели (**Find**) или осуществлять поиск необходимого файл (**Find File**), а также очищать окно команд (**Clear Command Window**), окно истории команд (**Clear Command History**) и содержимое рабочей области (**Clear Workspace**). Некоторые команды этого элемента меню могут обозначаться серым цветом, который свидетельствует о том, что команда в данный момент недоступна.

Элемент меню Debug, показанный на рис. 3.5.3-2c, содержит команды для работы с **m**-файлами, а также для отладки и редактирования этих файлов.

Элемент меню Desktop, изображенном на рис. 3.5.3-2e, размещает команды, которые позволяют отображать, скрывать, отделять и прикреплять оконные панели к

главному окну системы **MatLab**. Команда **Undock Command Window** отделяет окно панели от основного окна и размещает его в поле экрана. Команда **Desktop Layout** открывает подменю с дополнительным набором команд, предназначенные для отображения на рабочем столе предустановленных комбинаций оконных панелей. Команда **Default** возвращает внешний вид рабочего стола, который принят по умолчанию (рис. 3.5.3-1). Данное расположение основного окна системы **MatLab** является наиболее используемым при комплексной работе с системой **MatLab**. Команда **Command Window Only** удаляет с рабочего стола все панели, кроме **окна панели Command Window**. Данную особенность полезно использовать, если работа с системой **MatLab** осуществляется только с помощью командного окна. Команда **History and Command Window** служит для отображения на рабочем столе двух панелей: окна панели **Command History** - с левой стороны и окна панели **Command Window** - с правой стороны. Данная комбинация инструментальных средств на рабочем столе является более предпочтительной, чем одно окно команд, так как позволяет с помощью окна предыстории просматривать ранее введенные команды и при необходимости повторно выполнять различные последовательности этих команд. Команда **All Toolbar** отображает в основном окне шесть панелей: **Command Window, Current Directory, Workspace, Command History, Help и Profiler.**

Следующие шесть команд (**Command Window, Command History, Current Directory, Workspace, Help и Profiler)** управляют отображением соответствующих оконных панелей на экране.

Последние три команды – **Toolbar, Shortcuts Toolbar,** и **Titles** - позволяют разрешить или запретить отображение заголовков соответствующих элементов основного окна.

Элемент меню Window, представленном на рис. 3.5.3-2f, располагает командами, которые позволяют переключаться между различными оконными панелями.

Элемент меню Help, команды которого изображены на рис. 3.5.3-2g, обеспечивают доступ к справочной информации о системе **MatLab** и демонстрационным файлам по функциям этой системы. Кроме того, с помощью команды **Web Resources** можно перейти на соответствующую страницу сайта фирмы-разработчика системы **MatLab.**

Инструментальная линейка (Toolbar) дает наиболее простой и удобный способ работы с системой **MatLab**. Кнопки панели инструментов имеют изображение, явно подсказывающее их назначение (рис. 3.5.3-1). Для отображения подсказки о функции кнопки необходимо переместить курсор на кнопку и немного подождать.

3.5.3.1.2. Окно панели Command Window и простейшие вычисления

Система **MatLab** создана таким образом, что любые вычисления можно выполнять в режиме прямых вычислений, то есть без написания программы. Это превращает **MatLab** в необычайно мощный калькулятор, который способен производить не только обычные для калькуляторов вычисления (например, выполнять арифметические операции и вычислять элементарные функции), но и выполнять операции с векторами и матрицами, комплексными числами, рядами и полиномами и т.п. Можно почти мгновенно задать различные функции и построить их графики.

Командное окно панели **Command Window** используется для ввода команд и вывода результатов их выполнения. Работа с командным окном происходит в диалоговом режиме: пользователь вводит команду и передает ее ядру **MatLab**, ядро обрабатывает полученную команду и возвращает результат. Все команды вводятся в командную строку

после появления приглашения - **>>,** которое свидетельствует о готовности ядра системы **MatLab** к обработке очередной команды.

Таким образом, работа с системой в режиме прямых вычислений носит диалоговый характер и происходит по правилу «задал вопрос, получил ответ». То есть, пользователь набирает на клавиатуре, например, выражение, которое необходимо вычислить, редактирует его (если необходимо) в командной строке и завершает ввод нажатием клавиши **< Enter >**.

Рассмотрим простейший пример. Во-первых, вычислим результат выражения **2+3**. Для вычисления **2+3** необходимо ввести с клавиатуры в **Command Window.**

Command Window **Пример 3.5.3-1a**

```
>> 2+3
```

и нажать клавишу **<Enter>.** В итоге на следующей строке будет выведено:

Command Window **Пример 3.5.3-1b**

```
ans =
         5
>>
```

Далее о нажатии клавиши **<Enter>** упоминать не будем.

Из примера 3.5.3-1 видно, что **MatLab** по умолчанию создаёт переменную с именем **ans**, в которую записывает значение результата предыдущей операции (ее можно увидеть в рабочей области).

Рассмотрим второй пример.

Command Window **Пример 3.5.3-2a**

```
>> a=sin(pi/2)
a =
      1
>>
```

В этом примере создаётся переменная **a**, которой присваивается значение выражения **sin(pi/2)**, где **pi** – предопределённая в **MatLab** константа **π** (существует множество других предопределённых констант, например **e**, **i -** мнимая единица и др.).

Если после выражения поставить точку с запятой, то результат вычисления не будет выведен на экран, но переменная **a** будет создана и ей будет присвоено вычисленное значение:

Command Window **Пример 3.5.3-2b**

```
>> a=sin(pi/2);
>>
```

Значение этой переменной можно узнать, дважды щёлкнув по ней в рабочей области (**Workspace**), либо просто набрав её имя в командной строке.

Command Window **Пример 3.5.3-2c**

```
>>  a
a=
      1
>>
```

Переменную, которой ранее было присвоено значение, можно использовать для дальнейших вычислений, например:

Command Window **Пример 3.5.3-2d**

```
>>  x=3 - a
x =
      2
>>
```

Если в выражении указан операнд, значение которого неизвестно, **MatLab** выдает сообщение об ошибке:

Command Window **Пример 3.5.3-2e**

```
>>  (y+a)*(y-a)
??? Undefined function or variable 'y'.
>>
```

Если команда не помещается полностью в видимой части одной строки экрана, необходимо поставить три точки подряд, а затем нажать **<Enter>** и продолжать ввод команды на следующей строке.

Пусть, например, требуется найти значение выражения при **x = 0.2** и **y = –3.9:**

$$c = \sqrt{\frac{\sin(\frac{4}{3}\pi x) + e^{0.1y}}{\cos(\frac{4}{3}\pi x) + e^{0.1y}}} + \sqrt[3]{\frac{\sin(\frac{4}{3}\pi x) + e^{0.1y}}{\cos(\frac{4}{3}\pi x) + e^{0.1y}}}$$

Если набирать сразу все выражение, то получается достаточно длинная строка. Для переноса на следующую строку любой команды **MatLab** можно использовать знак переноса в виде трех подряд идущих точек. Вслед за знаком переноса необходимо нажать <**Enter**>:

Command Window **Пример 3.5.3-3a**

```
>>  x=0.2;
>>  y=-3.9;
>>  c=sqrt((sin(4/3*pi*x)+exp(0.1*y))/(cos(4/3*pi*x)+exp(0.1*y)))+...
((sin(4/3*pi*x)+exp(0.1*y))/(cos(4/3*pi*x)+exp(0.1*y)))^(1/3)
c =
        2.0451
>>
```

Однако проще всего решить эту задачу, используя промежуточные переменные.

Command Window **Пример 3.5.3-3b**

```
>>  x=0.2;
>>  y=-3.9;
>>  a=sin(4/3*pi*x)+exp(0.1*y);
>> b=cos(4/3*pi*x)+exp(0.1*y);
>> c=sqrt(a/b)+(a/b)^(1/3)
c =
       2.0451
>>
```

Здесь необходимо обратить внимание на некоторые важные особенности. Например, все операторы присваивания, кроме последнего, завершаются точкой с запятой для подавления вывода результата. Необязательно набирать выражение для **b**, похожее на только что введенное для **a**. После ввода третьей строки необходимо нажать клавишу **<↑>**. В командной строке появится предыдущее выражение, в которое следует внести необходимые изменения, а именно, необходимо заменить **sin()** на **cos(), b** на **a** и нажать <**Enter**>. Клавиши **<↑>** и **<↓>** служат для перехода по истории команд, то есть. для занесения ранее набранных команд в командную строку. Для быстрого перехода по истории команд можно также использовать окно панели **Command History**.

Необходимо обратить внимание, что в любой момент можно вывести значение переменной в командное окно, для чего следует набрать имя переменной в командной строке и нажать <**Enter**>, либо вызвать функцию **disp()**, например **disp(b)**.

Все переменные системы размещаются в рабочей области, содержимое которой (имена, размерность, тип) можно просмотреть командами **who** и **whos**.

Для очистки командного окна достаточно выполнить команду **clc,** а для очистки рабочей области – команду **clear.**

Необходимо иметь в виду, что любой фрагмент окна командной строки можно выделить и копировать в буфер, например, для переноса в **Word** или в командную строку. Возможен также перенос в командную строку текстовых фрагментов из других систем.

Все числовые значения, с которыми оперирует **MatLab,** в оперативной памяти компьютера представлены вещественными или комплексными значениями **double**. Это означает, что каждое вещественное число занимает **8 байт** в памяти и принимает по модулю значения из диапазона [10^{-308}, 10^{+308}], причем количество значащих десятичных цифр достигает **16**. Именно с такой точностью **MatLab** выполняет все вычисления.

Однако, при отображении числовых результатов на экране дисплея часть значащих цифр могут не отображаться в соответствии с установленным форматом вывода, причем отображаемые значения округляются по общепринятым в математике правилам.

Поскольку по умолчанию все вычисления в **MatLab** выполняются с двойной точностью, формат вывода может быть установлен двумя способами: как программным путем с помощью команды **format** в окне панели **Command Window**, так и с помощью установки соответствующих **свойств окна панели Command Window.**

Для установки **свойств среды** системы **MatLab,** а конкретно **свойств окна панели Command Window,** необходимо активизировать элемент основного меню **File,** а затем выбрать команду **Preference,** а в раскрывшемся диалоговом окне **Preferences** выделить вкладку **Command Window** (рис. 3.5.3-3).

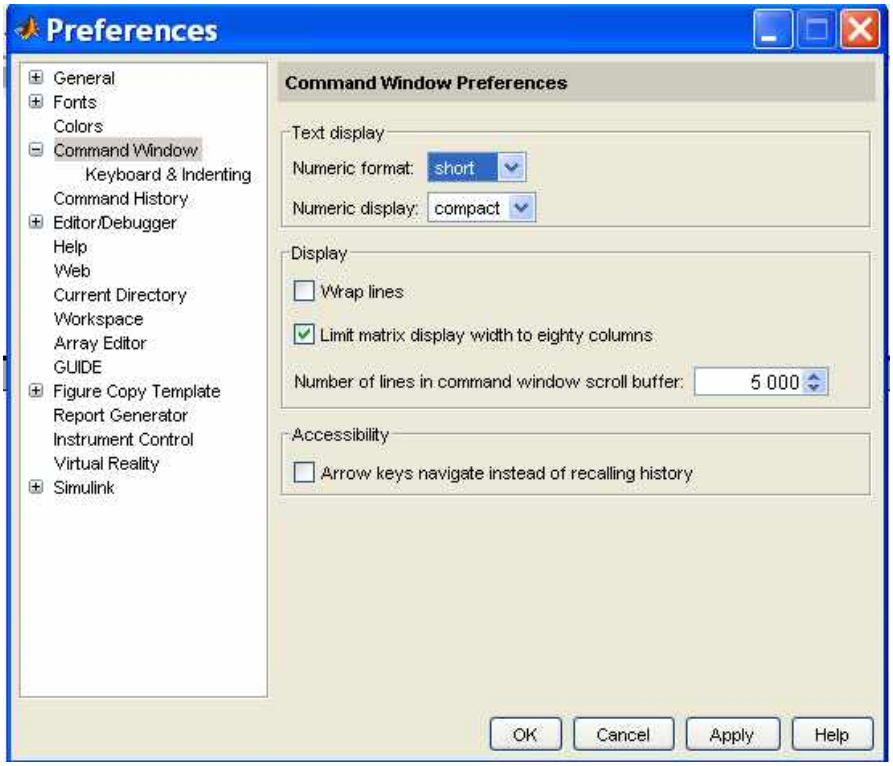


Рис. 3.5.3-3. Окно установки свойств среды системы **MatLab**

Внутри группы **Text display** окна **Preferences** будут расположены раскрывающиеся списки **Numeric format** и **Numeric display.** Далее из раскрывающегося списка **Numeric format** можно установить один из форматов, приведенных в табл. 3.5.3-1.

Таблица 3.5.3-1

Формат	Описание
short	Короткое число с плавающей точкой. Представляется с помощью четырех цифр после десятичной точки (по умолчанию)
long	Длинное число с плавающей точкой. Представляется с помощью четырнадцати цифр после десятичной точки
short e	Короткое число с плавающей точкой. Представляется с помощью пяти разрядов, четыре из которых отводится под вывод дробной части
long e	Длинное число с плавающей точкой. Представляется с помощью шестнадцати разрядов, пятнадцать из которых отводится под вывод дробной части
short g	Выбирается наилучшая форма представления числа из форматов short и short e
long g	Выбирается наилучшая форма представления числа из форматов long и long e
hex	Число выводится в шестнадцатеричной форме
+	Символьное обозначение числа: «+» - положительное число; «-» - отрицательное число; пробел - нулевое значение
rational	Число выводится в дробном виде

Для выделения результата вычисления или значения переменной **MatLab** вставляет пустую строку перед выводимым значением. Управлять появлением пустой строки или ее отсутствием можно в в раскрывшемся диалоговом окне **Preferences** внутри группы **Text display** с помощью следующих форматов:

- **compact** - строки с результатами выводятся подряд;
- **loose** - строки с результатами разделяются пустой строкой.

Команда **format** служит для установки формата из командной строки. Например, обращение

Пример 3.5.3-4

```
>>  format short e
>>
```

аналогично выбору короткого формата в окне **Preferences**.

Еще раз необходимо напомнить, что вне зависимости от установленного формата все вычисления производятся с двойной точностью, следовательно, после смены формата с **short** на **long** не требуется повторно находить значения переменных. Достаточно снова вывести их значения в командном окне.

Также еще раз напомним, что содержимое командной строки **MatLab** легко редактируется. Курсор можно перемещать с помощью стрелок **<←>, <→>** и удалять неправильно набранные символы с помощью клавиш <**Backspace**> или **<Delete>.** Как было уже сказано, удобным свойством системы является возможность использовать клавиши-стрелки **<↑>,<↓>** для доступа к стеку с ранее введенными командами. Таким образом, имеется возможность заново вызывать ранее вызванную команду, отредактировать ее и снова выполнить. Для небольших процедур это гораздо удобнее, чем писать и отлаживать специальные **m-файлы**, что требует постоянного перехода из окна **MatLab** в окно текстового редактора.

Все команды строчного редактора перечислены в таблице 3.5.3-2.

Таблица 3.5.3-2

Комбинация клавиш	Назначение
→	Перемещение курсора вправо на один символ
←	Перемещение курсора влево на один символ
Ctrl+→	Перемещение курсора вправо на одно слово
Ctrl+←	Перемещение курсора влево на одно слово
Home	Перемещение курсора в начало строки
End	Перемещение курсора в конец строки
↑ и ↓	Перелистывание предыдущих команд вверх или вниз для подстановки в строку ввода
Del	Стирание символа справа от курсора
Backspace	Стирание символа слева от курсора
Ctrl+k	Стирание до конца строки
Esc	Очистка строки ввода
Ins	Вкл/выкл режима вставки
PgUp	Перелистывание страниц сессии вверх
PgDn	Перелистывание страниц сессии вниз

3.5.3.1.3. Основные объекты системы MatLab

Математические выражения в **MatLab** стоятся, как и в большинстве языков программирования, из числовых констант, переменных, стандартных и нестандартных функций, соединенных знаками арифметических операций, например **+ ,- ,* , / ,^,** и круглых скобок. Кроме того, как было показано ранее, вид результата зависит от установленного формата.

Число – простейший объект языка **MatLab,** представляющий числовые данные. Числа могут быть представлены в целом, дробном, с фиксированной и плавающей точкой, а также в экспоненциальном виде. Например,

0, 2, -4, 4.67, 0.0005, 567.9e-7, 0.89e12.

Числа могут быть как действительными, так и комплексными**.** Комплексные числа содержат как действительные, так и комплексные чисти. В **MatLab** мнимая часть имеет множитель **i** или **j,** означающий корень квадратный из **-1.** Например,

3i, 5j, -5.1 +i8, 005e-5- j0.006.

Все операции над числами в **MatLab** выполняются в формате с двойной точностью. Однако, как нам уже известно, **MatLab** выдает числовые результаты в нормализованной форме с четырьмя цифрами после десятичной точки и одной до нее. Поэтому при работе с числовыми данными можно задавать различные форматы представления чисел.

Числовая константа – это предварительно определенное число (числовое значение). Числа (например, **1, -5, 3.97**) являются безымянными числовыми константами.

Системные константы (табл. 3.5.3-3) – это такие константы, значения которых задаются системой при загрузке. Однако по мере необходимости эти константы могут переопределяться.

Таблица 3.5.3-3

Константа	Назначение
i или j	Мнимая единица
pi	Число π=3.1415926…
eps	Погрешность вычислений над числами с плавающей точкой (по умолчанию 2^{-52})
realmin	Наименьшее число с плавающей точкой (по умолчанию 2^{-1022})
realmax	Наибольшее число с плавающей точкой (по умолчанию 2^{1022})
inf	Значение машинной бесконечности
NaN	Указание на нечисловой характер данных (Not-a-Number)

Символьная константа – это последовательность символов, заключенных в одиночные апострофы. Например:

```
'Кафедра ВМиП'
'88+0'
```

Комментарии в **MatLab** определяются с помощью символа **%**. Например,

```
% Это комментарий
```

Переменные – это имеющие имена объекты, способные хранить некоторые, разные по значению, данные. В зависимости от этих данных, переменные могут быть числовыми или символьными, векторными или матричными. При использовании переменных необходимо придерживаться правил:

- имя переменной (ее идентификатор) может состоять из символов латинского алфавита, знака подчёркивания и цифр, но начинаться обязательно с символа алфавита;
- прописные и строчные буквы различаются;
- пробел не входит в имя переменной.

Для задания переменным определенных значений используется операция ***присваивания***, обозначаемая знаком равенства **=**.

Command Window **Пример 3.5.3-5**

```
>>  a=3.25*(0.7-3.3/5.1)+2.3^3
a =
       12.3391
>> b=5*(2.2+3.9i)+0.8
b =
      11.8000 +19.5000i
>>
```

Простейшие арифметические операции системы **MatLab** приведены в табл. 3.5.3-4.

Таблица 3.5.3-4

Функция	Название	Операция	Синта-сис
plus	Плюс	**+**	**M1+M2**
uplus	Унарный плюс	**+**	**+M**
minus	Минус	**-**	**M1-M2**
uminus	Унарный минус	**-**	**-M**
mtimes	Матричное умножение	*****	**M1*M2**
times	Поэлементное умножение массивов	**.***	**A1.*A2**
mpower	Возведение матрицы в степень	**^**	**M^x**
power	Поэлементное возведение массива в степень	**.^**	**A.^x**
mldivide	Обратное (справа налево) деление матриц	****	**M1\M2**
mrdivide	Деление матриц слева направо	**/**	**M1/M2**
ldivide	Поэлементное деление массивов справа налево	**.**	**A1.\A2**
rdivide	Поэлементное деление массивов слево направо	**./**	**A1./A2**

Необходимо обратить внимание на то, что каждая операция имеет аналогичную по назначению функцию. Например, операции матричного умножения * соответствует функция **mtimes(Ml,M2)**. Примеры применения арифметических операций уже не раз приводились, так что ограничимся несколькими дополнительными примерами.

Command Window **Пример 3.5.3-6**

```
>>A=[1 2 3];
>>B=[4 5 6]:
>>B-A
 ans=
            3             3                 3
>>minus (B. A)
ans =
            3             3                 3
>>A. ^ 2
ans =
            1             4                 9
>> power(A,2)
ans =
            1             4                 9
>>A.\B
ans=
          4.0000        2.5000        2.0000
>>ldivide(A.B)
ans=
          4.0000        2.5000        2.0000
>>rdivide(A.B)
ans=
          0.2500        0.4000             0.5000
```

Соответствие функций операторам и командам в системе **MatLab** является одним из основных положений программирования в **MatLab**. Оно позволяет одновременно использовать элементы как операторного, так и функционального программирования.

Следует отметить, что в математических выражениях операции имеют определенный **приоритет исполнения**. Например, в **MatLab** приоритет логических операций выше, чем арифметических, приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше приоритета сложения и вычитания. Для изменения приоритета операций в математических выражениях используются круглые скобки.

MatLab обладает большим набором ***стандартных встроенных математических функций***. Некоторые из них приведены в табл. 3.5.3-5. При вызове математических функций аргумент заключается в круглые скобки. Полный список всех встроенных элементарных математических функций можно получить, набрав в командной строке.

Command Window **Пример 3.5.3-7**

```
>>help elfun
>>
```

Таблица 3.5.3-5

Тригонометрические функции (результат вычисляется в радианах)	
sin, cos, tan, cot	Синус, косинус, тангенс и котангенс
sec, csc	Секанс, косеканс
Обратные тригономе-кие функции (результат вычисляется в рад.)	
asin, acos,atan,atan2, acot	Арксинус, арккосинус, арктангенс и арккотангенс
asec, acsc	Арксеканс, арккосеканс
Гиперболические функции	
sinh, cosh, tanh, coth	Гипербол. синус, косинус, тангенс и котангенс
sech, csch	Гиперболические секанс и косеканс
asinh, acosh, atanh, acoth	Гиперболические арксинус, арккосинус, арктангенс и арккотангенс;
Экспоненциальная функция, логарифмы, степенные функции	
exp	Экспоненциальная функция
log, log2, log10	Логарифм натуральный, по основанию 2 и 10
sqrt	Квадратный корень
Модуль, знак и функции для работы с комплексными числами	
abs, sign	Модуль и знак числа
conj, imag, real	Комплексно-сопряжённое
mod, rem	Остаток от деления с учетом знака делимого и без
gcd, lcm	Наибольший и наименьший общий делитель
cell, fix, floor, round	округления

Команда **help** отображает в командном окне список разделов справочной системы. Для получения содержимого раздела необходимо указать через пробел его название после **help**, а для вывода детальной информации о какой-либо функции, следует ввести в строке с **help** ***имя функции***.

Возникающий в процессе вычислений комплексный результат не является ошибкой. **MatLab** автоматически переходит в область комплексных чисел, если это необходимо, продолжая вычисления. Например, если необходимо найти квадратный корень из **–1**. Более того, допустимы операции деления на ноль, которые приводят к стандартным переменным **Inf** или **–Inf**. Результат деления нуля на ноль есть **NaN** (Not a Number – не число).

Переполнение или потеря точности в **MatLab** при выполнении операций с числами с плавающей точкой не вызывает прекращение вычислений.

Запись комплексных величин, используемых в арифметических выражениях, напоминает общепринятые математические стандарты. Мнимые части комплексных чисел сопровождаются либо буквой **i**, либо буквой **j**:

Command Window **Пример 3.5.3-8**

```
>>x=1.5-0.5i
x =
      1.5000 - 0.5000i
>>y=1.5-0.5j
y =
      1.5000 - 0.5000j
>>
```

Если переменным **i** и **j** не присвоены какие-либо значения, то их можно использовать для формирования комплексных данных, используя знак умножения и располагая такой "сомножитель" до или после мнимой части:

Command Window **Пример 3.5.3-9**

```
>>a=1;
>>x=i*a
x =
      0 + 1.0000i
>>y=a*i
y =
      0 + 1.0000i
>>
```

Когда полные комплексные числа используются в операциях умножения, деления и возведения в степень, то для устранения неоднозначности их заключают в круглые скобки.

С помощью функции **real** и **imag** можно выделить вещественную и мнимую часть комплексного значения, а функция **complex** позволяет сконструировать комплексное значение по паре двух вещественных чисел.

Command Window **Пример 3.5.3-10**

```
>>b=5+j10;
>>c=real(d)
c=
     5
>>d=imag(b)
d=
     10
>>
```

Функцией **conj()** можно воспользоваться для получения комплексного сопряженного числа. Тот же результат можно получить, располагая апостроф вслед за комплексным значением.

Command Window **Пример 3.5.3-11**

```
>>t=conj(b)
t=
     5.0000e+000 -1.0000e+001i
>>g=b'
g=
     5.0000e+000 -1.0000e+001i
>>
```

Отметим, что над комплексными данными в **MatLab** определены все арифметические операции, а комплексные операнды и выражения могут использоваться в качестве аргументов стандартных функций.

Необходимо обратить внимание, что числовые переменные в **MatLab** являются двумерными массивами размера один на один. Представление всех данных в **MatLab** в виде массивов оказывается очень полезным, о чем подробнее будет сказано в следующих разделах.

3.5.3.1.4. Работа с векторами и матрицами

Пакет **MatLab** построена как система, ориентирующая на работу с матрицами, то есть все численные вычисления производятся в матричной форме. Система **MatLab** выполняет сложные и трудоемкие операции над векторами и матрицами даже в режиме прямых вычислений без какого-либо программирования. Ею можно пользоваться как мощнейшим калькулятором, в котором наряду с обычными арифметическими и алгебраическими действиями могут использоваться такие сложные операции, как инвертирование матрицы, вычисление ее собственных значений и принадлежащих им векторов, решение систем линейных уравнений, вывод графиков двумерных и трехмерных функций и многое другое. Важно отметить, что даже обычные числа и переменные в **MatLab** рассматриваются как матрицы размера **1 x 1**, что дает единообразные формы и методы проведения операций над обычными числами и массивами. Это также означает, что большинство функций может работать с аргументами в виде векторов и матриц. При необходимости вектора и матрицы преобразуются в массивы, и значения вычисляются для каждого их элемента.

Массивы являются одним из самых распространенных способов хранения данных и используются во всех языках программирования и математических пакетах. К особенностям работы с массивами в **MatLab** относится то, что одномерный массив может быть **вектор-строкой** или **вектор-столбцом**. Если способ представления массива важен, то мы будем подчеркивать, о строке или о столбце идет речь. Если же это несущественно, то будем говорить о **вектор-строках** и **вектор-столбцах** просто как о векторах или одномерных массивах. Напомним, что одномерный массив в **MatLab** есть двумерный, у которого ода из размерностей равна единице.

Для определения вектора используются квадратные скобки, а элементы вектора отделяются друг от друга:

- точкой с запятой, если требуется получить вектор–столбец;
- пробелом или запятой, если необходимо разместить элементы в векторе-строке.

Command Window **Пример 3.5.3-12**

```
>>a=[0.2; -3.9; 4.6]
a =
      0.2000
      -3.9000
      4.6000
>>b=[7.6; 0.1; 2.5]
b =
      7.6000
      0.1000
      2.5000
>>u=[0.1 0.5 -3.7 8.1]
u =
      0.1000  0.5000  -3.7000  8.1000
>>v=[5.2 9.7 3.4 -0.2]
v =
       5.2000   9.7000   3.4000   -0.2000
>>
```

Получить информацию о переменных, как мы уже знаем, можно с помощью окна **Workspace** или при помощи команды **whos.**

Для определения длины вектора используется функция **length(a)**, вектор **a** указывается в качестве ее входного аргумента.

Command Window **Пример 3.5.3-13**

```
>>L=length(a)
L =
       3
>>
```

Вектор-столбцы с одинаковым числом элементов можно складывать и вычитать друг из друга при помощи опреаций **"+" и "–".** Эти правила верны и для **вектор-строк**.

Сложение и вычитание **вектор-строк** и **вектор-столбцов** или векторов разных размеров приводит к ошибке. Операция * предназначена для умножения векторов по правилу матричного умножения. Поскольку **MatLab** различает **вектор-строки** и **вектор-столбцы**, то допустимо либо умножение **вектор-строки** на такой же по длине **вектор-столбец** (**скалярное произведение**), либо умножение **вектор-столбца** на **вектор-строку** (**внешнее произведение**, в результате которого получается **прямоугольная матрица**). **Скалярное произведение** двух векторов возвращает функция **dot()**, а **векторное** - **cross()**:

Command Window **Пример 3.5.3-14**

```
>>s=dot(a, b)
s=
       1.2630e+001
>>c=cross(a, b)
c=
       -1.0210e+001
       3.4460e+001
       2.9660e+001
>>
```

Для операции транспонирования зарезервирован символ апостроф - **'**. Если вектор содержит комплексные числа, то операция **'** приводит к комплексно-сопряженному вектору. При вычислении скалярного и векторного произведений функциями **cross()** и **dot()** не

обязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками. Результат получается верный, например, при обращении **c=cross(a,b')**, только **c** становится вектор-строкой.

MatLab поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения - **.*** (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Например, для векторов **a** и **b**, введенных выше, поэлементное умножение дает следующий результат:

Command Window **Пример 3.5.3-15**

```
>>c=a.* b
c =
       1.5200
       -0.3900
       11.5000
>>
```

Аналогичным образом работает поэлементное деление - **./** (точка с косой чертой). Кроме того, операция **.** (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения **a./b** и **b.\a** эквивалентны. Возведение элементов вектора **a** в степени, равные соответствующим элементам **b**, производится с использованием операции - **.^**. Для транспонирования **вектор-строк** или **вектор-столбцов** предназначено сочетание - **.'** (точка с апострофом). Операции **- '** и **.'** для вещественных векторов приводят к одинаковым результатам. Не обязательно применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции **a*2**, результат представляет собой вектор того же размера, что и **a**, с удвоенными элементами.

Векторы могут быть аргументами встроенных математических функций, таких, как **sin(), cos()** и т. д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

Command Window **Пример 3.5.3-16**

```
>>q=sin([0 pi/2 pi])
q =
       0   1.0000   0.0000
>>
```

Однако для вычисления более сложной функции от вектора значений, скажем, $\frac{\alpha Sin\alpha + \alpha^2}{\alpha + 1}$, где **α,** например**,** является вектором-строкой, состоящей из четырех элементов, выражение **f=(α*sin(α)+α^2)/(α+1)** вызовет ошибку уже при попытке умножения **α** на **sin(α)**. Дело в том, что **α** является **вектор-строкой** длиной четыре, то есть хранится в двумерном массиве размером один на четыре. Точно также представлена и функция **sin(α)**, следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла. Аналогичная ситуация возникает и при возведении вектора **α** в квадрат, то есть, фактически, при вычислении **α*α.**

Правильная запись выражения в **MatLab** требует использования поэлементных операций.

Command Window **Пример 3.5.3-17**

```
>>f=(a.*sin(a)+a.^2)./(a+1);
>>
```

Часто требуется вычислить функцию от вектора значений аргумента, отличающихся друг от друга на постоянный шаг. Для создания таких **векторов-строк** предусмотрена операция двоеточие - **:**, которая отделяет начальное значение аргумента, шаг и конечное значение аргумента.

Command Window **Пример 3.5.3-18**

```
>>x=-1.2:0.5:1.8
x =
      -1.2000  -0.7000  -0.2000  0.3000  0.8000  1.3000  1.8000
>>f=(x.*sin(x)+x.^2)./(x+1)
f =
      -12.7922  3.1365   0.0997   0.1374   0.6744   1.2794   1.7832
>>
```

Шаг может быть отрицательным, в этом случае начальное значение должно быть больше, либо равно конечному значению для получения непустого вектора. Если шаг равен единице, то его можно не указывать, например:

Command Window **Пример 3.5.3-17**

```
>>n=-3:4
n =
       -3 -2 -1 0 1 2 3 4
>>
```

Ясно, что для заполнения вектор-столбца элементами с постоянным шагом следует транспонировать вектор-строку.

Создание векторов при помощи двоеточия и умение производить поэлементные операции необходимо для визуализации массивов данных.

Command Window **Пример 3.5.3-18**

```
>>x=10:-2:0
x =
       10         8         6         4         2                0
>>sin(x)=
       -0.5440    0.9894   -0.2794   -0.7568    0.9093          0
>>
```

Необходимо отметить, что при умножении векторов друг на друга получается скаляр.

Command Window **Пример 3.5.3-19**

```
>>y=[2 3 5 8 3 9];
>>x*y
??? Error using ==> *
  Inner matrix dimensions must agree.
>>
```

Здесь допущена ошибка, так как матрицы (векторы) не согласованны. Правильно будет:

Command Window **Пример 3.5.3-20a**

```
>>x*y'
ans =
          112
>>
```

или можно было создать **y** как матрицу-столбец:

Command Window **Пример 3.5.3-20b**

```
>>y=[2;3;5;8;3;9]
>>
```

MatLab обладает большим набором встроенных функций для обработки векторных данных, часть из них приведена в табл. 3.5.3-6.

Таблица. 3.5.3-6

Функции	Назначение
s=sum(a)	Сумма всех элементов вектора **a**
p=prod(a)	Произведение всех элементов вектора **a**
m=max(a)	Нахождение максимального значения среди элементов вектора **a**
[m,k]=max(a)	Второй выходной аргумент **k** содержит номер максимального элемента в векторе **a**
m=min(a)	Нахождение минимального значения среди элементов вектора **a**
[m,k]=min(a)	Второй выходной аргумент **k** содержит номер минимального элемента в векторе **a**
m=mean(a)	Вычисление среднего арифметического элементов вектора **a**
a1=sort(a)	Упорядочение элементов вектора **a** по возрастанию
[a1,ind]=sort(a)	Второй выходной аргумент **ind** является вектором из целых чисел от **1** до **length(a)**, который соответствует проделанным перестановкам

Полный список имеющихся функций выводится в командное окно при помощи команды **help datafun**, а для получения подробной информацию о каждой функции требуется указать ее имя в качестве аргумента команды **help.** Обратите внимание на то, что ряд функций допускает обращение к ним как с одним, так и с двумя выходными аргументами. В случае нескольких выходных аргументов они заключаются в квадратные скобки и отделяются друг от друга запятой.

Очень часто требуется обработать только часть вектора, или обратиться к некоторым его элементам. Разберем правила **MatLab**, по которым производится индексация векторных данных. Для доступа к элементу вектора необходимо указать его номер в круглых скобках сразу после имени переменной, в которой содержится вектор. Например, сумма первого и третьего элементов вектора **y** находится при помощи выражения

Command Window **Пример 3.5.3-21**

```
>>s=y(1)+y(3);
>>
```

Обращение к последнему элементу вектора можно произвести с использованием аргумента **end,** то есть **y(end)** и **y(length(v))** приводят к одинаковым результатам.

Указание номеров элементов вектора можно использовать и при вводе векторов, последовательно добавляя новые элементы (не обязательно в порядке возрастания их номеров).

Command Window **Пример 3.5.3-22**

```
>>h=10;
>>h(2)=20;
>>h(4)=40
h =
       10 20 0 40
>>
```

Заметим, что для ввода первого элемента **h** не обязательно указывать его индекс, так как при выполнении оператора **h=1** создается вектор (массив размера один на один). Следующие операторы присваивания приводят к автоматическому увеличению длины вектора **h**, а пропущенные элементы (в нашем случае **h(3)**) получают значение ноль.

Индексация двоеточием позволяет выделить идущие подряд элементы в новый вектор. Начальный и конечный номера указываются в круглых скобках через двоеточие, например:

Command Window **Пример 3.5.3-23**

```
>>z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
>>znew=z(3:6)
znew =
            7.9000 4.5000 7.2000 -8.1000
>>
```

Применение встроенных функций обработки данных к некоторым последовательно расположенным элементам вектора не представляет труда. Следующий вызов функции **prod()** вычисляет произведение элементов вектора **z** со второго по шестой:

Command Window **Пример 3.5.3-24**

```
>>p=prod(z(2:6))
>>
```

Индексация векторов служит для выделения элементов с заданными индексами в новый вектор. Индексный вектор должен содержать номера требуемых элементов, например:

Command Window **Пример 3.5.3-25**

```
>>ind=[3 5 7];
>>znew=z(ind)
znew =
              7.9000 7.2000 3.4000
>>
```

Для нахождения суммы элементов произвольного вектора **z** с четными индексами необходимо выполнить следующие команды:

Command Window **Пример 3.5.3-26**

```
>>ind=2:2:length(z);
>>s=sum(z(ind))
>>
```

Конструирование новых векторов из элементов имеющихся векторов производится при помощи квадратных скобок. Следующий оператор приводит к образованию вектора, в котором пропущен пятый элемент вектора **z.**

Command Window **Пример 3.5.3-27**

```
>>znew=[z(1:4)   z(6:end)]
znew =
              0.2000 -3.8000 7.9000 4.5000 -8.1000 3.4000
>>
```

Для определения матрицы необходимо задать значение элементов строк и разделить строки матрицы символом - ;, заключенными в квадратные скобки:

A=[v1;v2;v3], где **v1, v2, v3** -векторы одинаковой размерности.

Кроме того для определения матриц существует множество функций. Некоторые из них приведены в табл. 3.5.3-7.

Таблица. 3.5.3-7

Функция	Результат и примеры вызовов
zeros	Нулевая матрица **F=zeros(4,5) F=zeros(3) F=zeros([3 4])**
eye	Единичная прямоугольная матрица (единицы расположены на главной диагонали) **I=eye(5,8) I=eye(5) I=eye([5 8])**
ones	Матрица, целиком состоящая из единиц **E=ones(3,5) E=ones(6) E=ones([2 5])**
rand	Матрица, элементы которой — случайные числа, равномерно распределенные на интервале (0,1) **R=rand(5,7) R=rand(6) R=rand([3 5])**
randn	Матрица, элементы которой — случайные числа, распределенные по нормальному закону с нулевым средним и дисперсией, равной единице N=**randn(5,3) N=randn(9) N=randn([2 4])**
diag	1) диагональная матрица, элементы которой задаются во входном аргументе – векторе **D=diag(v)** 2) диагональная матрица со смещенной на **k** позиций диагональю (положительные k — смещение вверх, отрицательные — вниз), результатом является квадратная матрица размера **length(v)+abs(k) D=diag(v,k)** 3) выделение главной диагонали из матрицы в вектор **d=diag(A)** 4) выделение **k**-ой диагонали из матрицы в вектор **d=diag(A,k)**

Рассмотрим примеры формирования матриц.

Command Window **Пример 3.5.3-28**

```
>>Z=zeros(3,5)
Z =
      0     0     0     0     0
      0     0     0     0     0
      0     0     0     0     0
>>A=[1 2 3 4; 5 6 7 8; 9 10 11 12] ;    %матрица A(4x3)
A =
        1     2     3     4
        5     6     7     8
        9    10    11    12
>>
```

Причем **элементами** матрицы могут быть **матрицы** или **вектора**.

Command Window **Пример 3.5.3-29**

```
% первая "строка" матрицы  K-матрица A,
% вторая - вектор  [1 2 3 4]
>>K=[A;1 2  3]
K =
        1     2     3     4
        5     6     7     8
        9    10   11    12
        1     2     3     4
>>
```

Таким образом, как нам уже известно, вектор является матрицей с числом строк или столбцов равным **1,** а число – матрица из одного элемента:

Command Window **Пример 3.5.3-30**

```
>>x=[1 2 3 4]
x =
      1     2     3     4
>>c=1.256
c =   1.2560
>>
```

Над матрицами и элементами матрицы можно выполнять различные операции, но при этом необходимо учитывать правила работы с матрицами. Рассмотрим примеры этих операций.

1)Умножение матрицы **A** на матрицу **B**:

Command Window **Пример 3.5.3-31**

```
>>B=[2 5; 6 4; 6 5; 8 3]
B =
      2     5
      6     4
      6     5
      8     3
>>C=A*B
C =
       64    40
      152   108
      240   176
>>
```

2)Умножение соответствующих элементов матриц необходимо использовать оператор поэлементного доступа "**.**":

Command Window **Пример 3.5.3-32**

```
>>A1=[1 3 4 6; 9 7 4 0; 8 6 3 9];
>>C2=A.*A1
C2 =
         1     6    12    24
        45    42    28     0
        72    60    33   108
>>
```

3) Добавление элементов ("увеличение матрицы"):

Command Window **Пример 3.5.3-33**

```
>>a1=[A;x]
a1 =
         1     2     3     4
         5     6     7     8
         9    10    11    12
         1     2     3     4
>>
```

Пара квадратных скобок при этом является оператором объединения.

4)Транспонирование матрицы (апостроф - **'**):

Command Window **Пример 3.5.3-34**

```
>>Bt=B'
Bt =
        2     6     6     8
        5     4     5     3
>>
```

5)Удаление строки (столбца матрицы), в примере удален второй столбец:

Command Window **Пример 3.5.3-35**

```
>>A(:,2)=[ ]      %круглые скобки – оператор извлечения
A =
       1     3     4
       5     7     8
       9    11    12
>>
```

6) Извлечение элемента матрицы:

Command Window **Пример 3.5.3-36**

```
>>y=A(3,3)
%обращение к элементу в 3-й строке и 3-м столбце
y =
      12
>>  %    или
>>  A(5)  %обращение по номеру так,
%если записать все столбцы матрицы друг за другом>
ans =    7
```

7) Извлечение строк или столбцов матрицы:

Для извлечения строк или столбцов вместо номера элементов в строке или столбце используется двоеточие, которое является оператором перечисления (с первого до последнего элемента строки или столбца):

Command Window **Пример 3.5.3-37**

```
>>%извлечение 3-й строки из A,
>>%двоеточие означает весь набор чисел
>>y=A(3,:)
y =
       9    11    12
>>% в этом примере это элементы из  3-й стоки и 1 2 3
столбцов.
```

Двоеточие, поставленное между двумя числами – это перечисление от одного до другого числа, с указанным интервалом (по умолчанию **1**):

Command Window **Пример 3.5.3-38**

```
>>x=0:5          %вектор x = (0 1 2 3 4 5)
x =
       0     1     2     3     4     5
>>x=0:0.1:0.5    %вектор x = (0 0,2 0,3 0,4 0,5)
x =
       0    0.1000    0.2000    0.3000    0.4000    0.5000
>>
```

Таким образом, матрицы небольших размеров удобно вводить из командной строки. Существует три способа ввода матриц. Например, необходимо определить следующую матрицу:

$$A = \begin{bmatrix} 0.7 & -2.5 & 9.1 \\ 8.4 & 0.3 & 1.7 \\ -3.5 & 6.2 & 4.7 \end{bmatrix}$$

Первый способ предполагает набрать в командной строке (разделяя элементы строки матрицы пробелами): **A=[0.7 –2.5 9.1** и нажать **<Enter>**. Курсор перемещается в следующую строку (символ приглашения командной строки **>>** отсутствует). Элементы каждой следующей строки матрицы набираются через пробел, а ввод строки завершается нажатием на **<Enter>**. При вводе последней строки в конце ставится закрывающая квадратная скобка:

Command Window **Пример 3.5.3-39a**

```
>>A=[0.7 -2.5 9.1
8.4 0.3 1.7
-3.5 6.2 4.7]
A =
      0.7000   -2.5000    9.1000
      8.4000    0.3000    1.7000
     -3.5000    6.2000    4.7000
>>
```

Второй способ ввода матрицы основан на том, что матрицу можно рассматривать как вектор-столбец, каждый элемент которого является строкой матрицы. Поскольку точка с запятой используется для разделения элементов вектора-столбца, то ввод, к примеру, матрицы

$$B = \begin{bmatrix} 6.1 & 0.3 \\ -7.9 & 4.4 \\ 2.5 & -8.1 \end{bmatrix}$$ осуществляется оператором присваивания:

Command Window **Пример 3.5.3-39b**

```
>>B=[6.1 0.3; -7.9 4.4; 2.5 -8.1]
B =
     6.1000    0.3000
    -7.9000    4.4000
     2.5000   -8.1000
>>
```

Очевидно, что допустима такая трактовка матрицы, при которой она считается вектор-строкой, каждый элемент которой является столбцом матрицы. Следовательно, для ввода матрицы $C = \begin{bmatrix} 0.4 & -7.2 & 5/3 \\ 0.1 & -2.1 & -9.5 \end{bmatrix}$ достаточно воспользоваться командой:

Command Window **Пример 3.5.3-40**

```
>>  C=[[0.4; 0.1] [-7.2; -2.1] [5.3; -9.5]]
C =
      0.4000   -7.2000    5.3000
      0.1000   -2.1000   -9.5000
>>
```

Необходимо обратить внимание, что внутренние квадратные скобки действительно нужны. Оператор **C=[0.4; 0.1 -7.2; -2.1 5.3; -9.5]** является недопустимым и приводит к

сообщению об ошибке, поскольку оказывается, что в первой строке матрицы содержится только один элемент, во второй и третьей — по два, а в четвертой - снова один.

Еще можно воспользоваться командой **whos** для получения информации о переменных **A, B** и **C** рабочей среды. В командное окно выводится таблица с информацией о размерах массивов, памяти, необходимой для хранения каждого из массивов, и типе – **double array**:

Command Window **Пример 3.5.3-41**

```
>>whos A B C
Name Size Bytes Class
A        3x3  72      double array
B        3x2 48       double array
C        2x3 48       double array
>>
```

Функция **size()** позволяет установить размеры массивов, она возвращает результат в виде вектора, первый элемент которого равен числу строк, а второй – столбцов:

Command Window **Пример 3.5.3-42**

```
>>s=size(B)
s =
     3 2
>>
```

Сложение и вычитание матриц одинаковых размеров производится с использованием знаков **+, -.** Звездочка ***** служит для вычисления матричного произведения, причем соответствующие размеры матриц должны совпадать.

Допустимо умножение матрицы на число и числа на матрицу, при этом происходит умножение каждого элемента матрицы на число и результатом является матрица тех же размеров, что и исходная. Апостроф - **'** предназначен для транспонирования вещественной матрицы или нахождения сопряженной к комплексной матрице. Для возведения квадратной матрицы в степень применяется знак **^**.

MatLab обладает многообразием различных функций и способов для работы с матричными данными. Для обращения к элементу двумерного массива следует указать его номер строки и номер столбца в круглых скобках после имени массива.

Индексация двоеточием позволяет получить часть матрицы — строку, столбец или блок, например:

Command Window **Пример 3.5.3-43**

```
>>c1=A(2:3,2)
c1 =
        0.3000
        6.2000
>> r1=A(1,1:3)
r1 =
        0.7000 -2.5000 9.1000
>>
```

Для обращения ко всей строке или всему столбцу не обязательно указывать через двоеточие начальный (первый) и конечный индексы, то есть операторы **r1=A(1,1:3)** и **r1=A(1,:)** эквивалентны. Для доступа к элементам строки или столбца от заданного до последнего можно использовать **end,** так же как и для векторов: **A(1,2:end)**. Выделение

блока, состоящего из нескольких строк и столбцов, требует индексации двоеточием, как по первому измерению, так и по второму.

Пусть в массиве **T** хранится матрица:

$$T = \begin{bmatrix} 1 & 7 & -3 & 2 & 4 & 9 \\ 0 & -5 & -6 & 3 & -8 & 7 \\ 2 & 4 & 5 & -1 & 0 & 3 \\ -6 & -4 & 7 & 2 & 61 & \end{bmatrix}$$

Для выделения элементов матрицы **T** со второй строки по третью и со второго столбца по четвертый, достаточно использовать оператор:

Command Window **Пример 3.5.3-44**

```
>>c1=A(2:3,2)
c1 =
          0.3000
          6.2000
>>
```

Ранее было описано применение поэлементных операций к векторам. Поэлементные вычисления с матрицами производятся практически аналогично, разумеется, необходимо следить за совпадением размеров матриц:

- **A.*B, A./B** - поэлементные умножение и деление;
- **A.^p** - поэлементное возведение в степень, **p** - число;
- **A.^B** - возведение элементов матрицы **A** в степени, равные соответствующим элементам матрицы **B**;
- **A.'** - транспонирование матрицы (для вещественных матриц **A'** и **A.'** приводят к одинаковым результатам).

Иногда требуется не просто транспонировать матрицу, но и "развернуть" ее. Разворот матрицы на 90^0 против часовой стрелки осуществляет функция **rot90( )**:

Command Window **Пример 3.5.3-45**

```
>>Q=[1 2;3 4]
Q =
       1 2
       3 4
>>R=rot90(Q)
R =
       2 4
       1 3
>>
```

Допустимо записывать сумму и разность матрицы и числа, при этом сложение или вычитание применяется, соответственно, ко всем элементам матрицы. Вызов математической функции от матрицы приводит к матрице того же размера, на соответствующих позициях которой стоят значения функции от элементов исходной матрицы.

В **MatLab** определены и матричные функции, например, **sqrtm( )** предназначена для вычисления квадратного корня.

Например, найдем квадратный корень из матрицы $k = \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$ и проверим полученный результат, возведя его в квадрат (по правилу матричного умножения, а не поэлементно):

Command Window **Пример 3.5.3-46**

```
>>K=[3 2; 1 4];
>> S=sqrtm(K)
S =
       1.6882 0.5479
       0.2740 1.9621
>>S*S
ans =
          3.0000 2.0000
          1.0000 4.0000
>>
```

Матричная экспонента вычисляется с использованием **expm()**. Специальная функция **funm()** служит для вычисления произвольной матричной функции.

Все функции обработки данных могут быть применены и к двумерным массивам. Основное отличие от обработки векторных данных состоит в том, что эти функции работают с двумерными массивами по столбцам, например, функция **sum()** суммирует элементы каждого из столбцов и возвращает вектор-строку, длина, которой равна числу столбцов исходной матрицы:

Command Window **Пример 3.5.3-47**

```
>>M=[1 2 3; 4 5 6; 7 8 9]
M =
         1 2 3
         4 5 6
         7 8 9
>>s=sum(M)
s =      12 15 18
```

Если в качестве второго входного аргумента **sum()** указать **2**, то суммирование произойдет по строкам. Для вычисления суммы всех элементов матрицы требуется дважды применить **sum()**:

Command Window **Пример 3.5.3-48**

```
>>s=sum(sum(M))
s =       45
>>
```

Очень удобной возможностью **MatLab** является конструирование матрицы из матриц меньших размеров. Пусть заданы матрицы:

$$M1=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\quad M2=\begin{bmatrix}-2 & -3 & -5\\ -1 & -5 & -6\end{bmatrix}\quad M3=\begin{bmatrix}9 & 8\\ -7 & -5\\ 1 & 2\end{bmatrix}\quad M4=\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Требуется составить из **M1, M2, M3**и **M4** блочную матрицу **M**

$$M=\left[\begin{array}{c|c} M1 & M2\\ \hline M3 & M4\end{array}\right].$$

Можно считать, что **M** имеет размеры два на два, а каждый элемент является, соответственно, матрицей **M1, M2, M3** или **M4**. Следовательно, для получения в рабочей среде **MatLab** матрицы **M** требуется использовать оператор:

Command Window **Пример 3.5.3-49**

```
>>M=[M1 M2; M3 M4];
>>
```

После ввода или формирования элементов векторов и матриц могут возникнуть ошибки. Для контроля и исправления отдельных элементов векторов и матриц можно воспользоваться окном редактора данных **Array Editor.**

Окно **Array Editor** (окно редактора массива данных) состоит из панели инструментов и области просмотра значений переменных. В случае откреплённого от рабочего стола окна редактирования данных в окне присутствует главное меню, которое совпадает с главным меню рабочего стола, и строка состояния (рис. 3.5.3-4). В окне редактора данных можно отображать несколько переменных. Переключение между переменными реализуется **Array Editor**. На рис. 3.5.3-4 в окне редактирования данных находится переменная **d**.

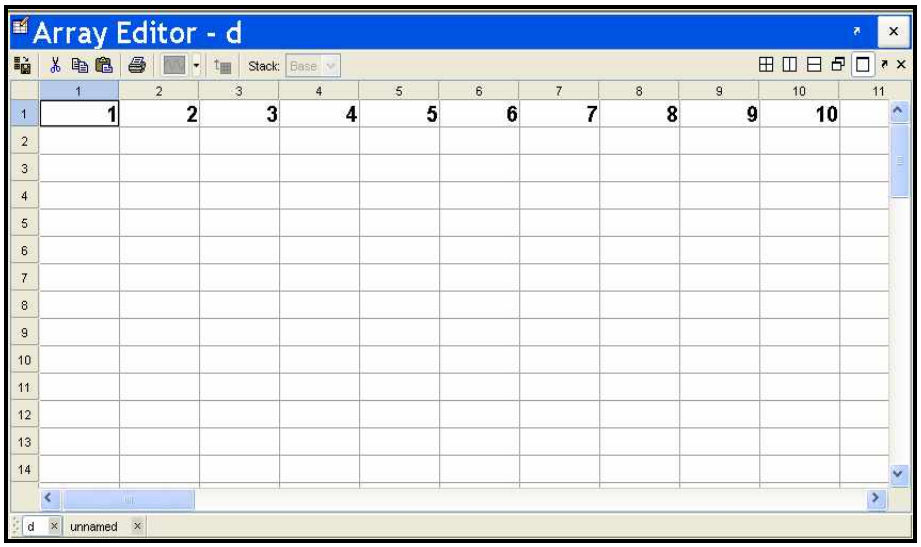


Рис. 3.5.4-4. Окно редактора данных

Панель инструментов окна редактирования данных используется для записи данных (**Save**), для перемещения (**Cut**) и копирования (**Copy**) в буфер обмена выделенных значений, вставки (**Paste**) значений из буфера и печати (**Print**).При перемещении выделенных значений в буфер обмена на их месте будут записаны нули. Количество и размерность числовых значений при вставке из буфера обмена должны совпадать с областью, выделенной для вставки.

3.5.3.1.5. Окна Workspace и Окно Command History

Как было сказано ранее, основное меню является контекстно-зависимым. Поэтому при активном окне **Workspace** элементы основного меню (рис. 3.5.3-5) будут отличаться от элементов при активном окне рабочей области.

Как видно из рис. 3.5.3-5 здесь появились дополнительные элементы:

View - содержит команды главного меню для отображения в окне **Workspace** различной информации и ее сортировки;

Graphics - содержит команды главного меню для работы с графическим окном.

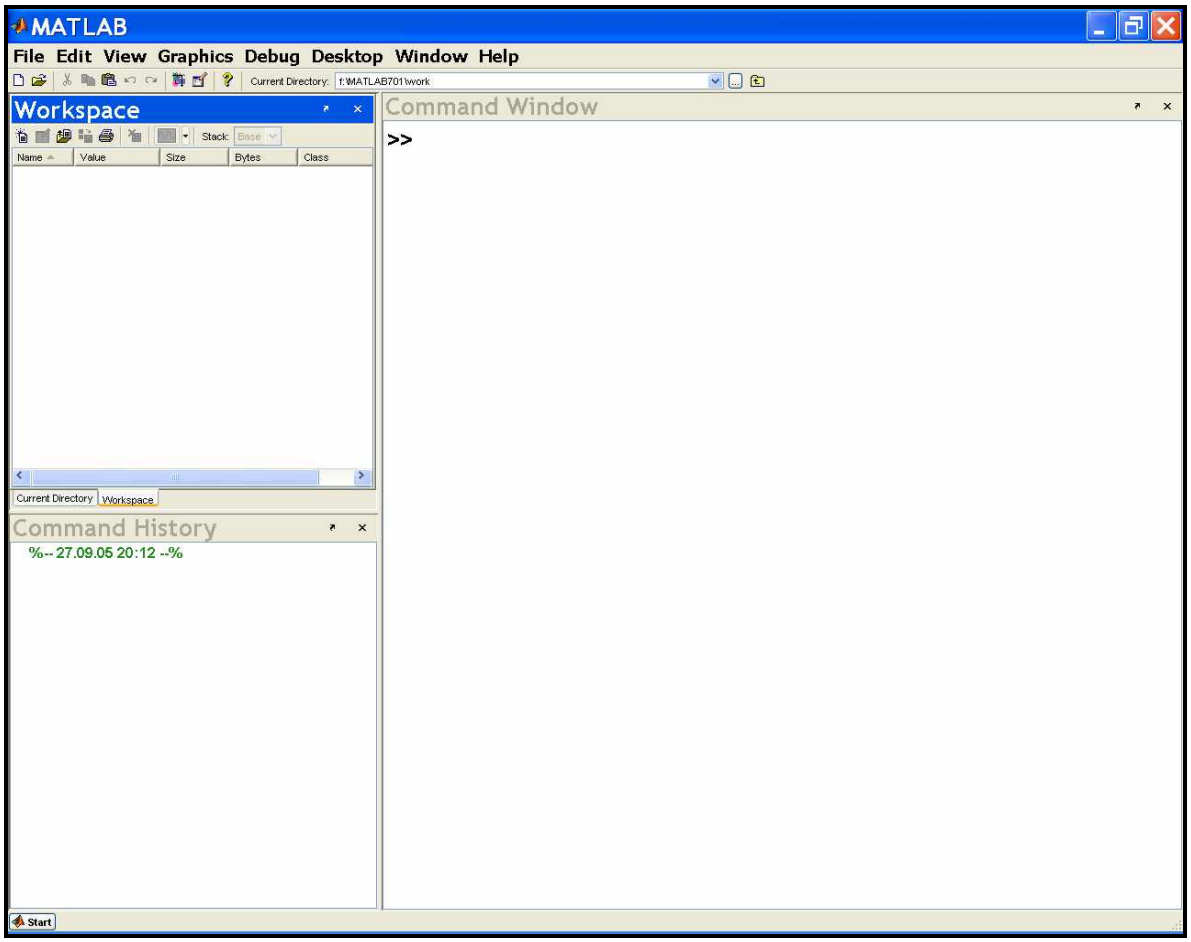


Рис. 3.5.3-5. **Основное окно** системы **MatLab** при активном окне **Workspace**

Команды элементов основного меню **View** и **Graphics** показаны на рис. 3.5.4-6.

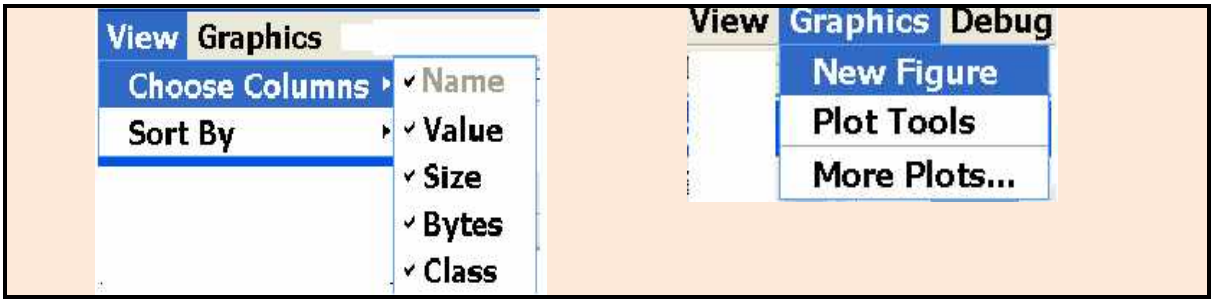


Рис. 3.5.3-6. Команды элементов главного меню системы **MatLab** при активном окне **Workspace**

Окно **Workspace** (рис. 3.5.3-6), предназначено для быстрого просмотра атрибутов переменных, располагающихся в рабочей области. С помощью окна **Workspace** можно увидеть имя переменной (**Name**), значение (**Value**), ее размер (**Size**), число байтов (**Bytes**), занимаемых переменной в памяти, и ее класс (**Class**). Для идентификации класса переменной слева от имени используется соответствующая иконка.

Окно **Workspace** можно отобразить на экране либо с помощью соответствующей команды меню **Desktop,** либо с помощью задания команды **Workspace** в командном окне. С помощью элементов основного меню **View** можно изменять внешний вид окна **Workspace** (скрывать или показывать поля **Size**, **Value**, **Bytes** и **Class**), а также сортировать переменные по имени, размерности, количеству байт

и классу. Более быстрый способ сортировки переменных по атрибутам реализуется щелчком левой кнопки мыши на имени соответствующего атрибута.

Строка инструментов окна **Workspace** позволяет выполнять следующие команды: создание новой переменной (**New variable**); открытие редактора данных с целью просмотра или редактирования значений выделенной переменной (**Open Selection**); загрузка данных из файла в рабочую область (**Load Data File**); сохранение рабочей области в файле (**Save**); печать содержимого рабочей области (**Print**); удалять выделенную переменную (**Delete**); построение различных типов графиков (**plot(d)**); переключение с помощью списка **Stack** между основной рабочей областью (**Base)** и рабочей областью функций во время их отладки (**Stack**).

Просматривать содержимое рабочей области, загружать и удалять данные можно не только с помощью средств окна **Workspace,** но также с помощью **Command Window.** Для просмотра имен переменных, находящихся в рабочей области, необходимо в командную строку ввести команду **who**. Результат задания команды **who** соответствует внешнему виду окна **Workspace** при отключенных полях **Size, Bytes** и **Class**. Полную информацию о содержании рабочей области можно получить с помощью команды: **whos**. В результате информация, выведенная в командное окно, будет соответствовать представленной в окне просмотра рабочей области при всех включенных полях, с тем исключением, что в командное окно будет выведено общее количество переменных и общий объем занимаемой ими памяти. Для удаления переменной из рабочей области необходимо ввести в командную строку команду

Command Window **Пример 3.5.3-50**

```
>>clear имя_переменной
>>
```

Очистка рабочей области осуществляется с помощью команды **clear** без параметров. Загрузка всех данных из файла реализуется командой **load** *имя_файла*, а выборочная загрузка - командой **load** *имя_файла имя__переменной***.**

Для сохранения рабочей области на диске необходимо ввести команду **save** *имя_файла***.**

Данные будут сохранены в файле с расширением **.mat**. Выборочное сохранение переменных из рабочей области обеспечивается командой **save** *имя_файла имя_переменной***.**

Редактор данных, изображенный на рис 3.5.3-4, предназначен для просмотра и редактирования значений переменных. Под редактированием переменных подразумевается не только изменение значений элементов массива, но также и изменение размера массива.

Редактор данных вызывается двойным щелчком на имени переменной в окне просмотра рабочей области или заданием в командном окне команды **openvar ('** *имя_переменной***').**

Окно истории команд, показанное на рис. 3.5.3-1 служит для просмотра команд, заданных ранее в командной строке **Command Window.** В окне истории команд можно также просмотреть дату и время начала сеанса работы с системой **MatLab**. Сеанс работы с системой **MatLab** начинается после ее загрузки в память и вывода на экран ее рабочего стола. Завершение сеанса работы сопровождается закрытием основного окна системы **MatLab**.

С помощью контекстного меню окна истории команд, изображенного на рис. 3.5.3-7 можно выполнять следующие действия: вырезать (**Cut**) и копировать (**Copy**) выделенные строки в буфер обмена; повторно выполнять команду или серию выделенных команд (**Evaluate Selection**); создавать новый **m**-файл (**Create M-File**) и **Shortcut** путем копирования выделенных строк; а также удалять выделенные строки (**Delete Selection**); удалять все строки из окна истории

команд до выделенной строки (**Delete to Selection**) и полностью очищать окно истории команд (**Delete Entire History**).

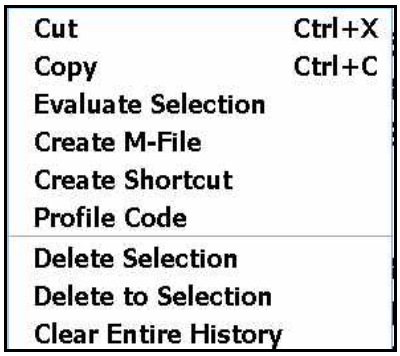


Рис. 3.5.3-7. Контекстное меню окна истории команд

Изменять параметры работы окна истории команд можно с помощью диалогового окна **Preferences**, представленного на рис. 3.5.3-3, при активном инструментальном средстве **Command History**. Диалоговое окно вызывается из меню **File** командой **Preferences...** В окне присутствуют две группы опций: **Settings** (Настройки) и **Saving** (Сохранение). В первой группе опция включается установкой флажка напротив ее имени. Во второй группе опции переключаются при щелчке на соответствующем имени.

В группе **Settings** можно включать или выключать следующие опции: **Save exit/quit command** – сохранять в истории команду **exit/quit** (при задании одной из этих команд система **MatLab** завершает работу); **Save consecutive duplicate commands** – сохранять одинаковые команды, заданные в командном окне друг за другом; **Save commands entered at an input prompt** – сохранять команды, введенные в командную строку в ответ на команду **input**.

Command Window **Пример** 3.5.4-51

```
>>  n=input (' Введите номер варианта')};
Allow Drag and Drop editing
>>
```

В группе **Saving** можно переключаться между следующими опциями: **Save history file on quit** – сохранять файл истории при выходе (история команд сохраняется в файле **history.m**); **Save after n commands** – сохранять файл истории после ввода каждой **n** команды; **Don't save history file** – не сохранять файл истории команд (несмотря на включенную опцию, в течение сеанса работы в окне истории команд сохраняются все команды, которые будут удалены при выходе).

Окно **Command History** хранит все команды, набираемые пользователем. В отличие от содержимого **Command Window** сюда не попадают сообщения системы и результаты вычислений.

3.5.3.2. Построение графиков и визуализация результатов вычислений

3.5.3.2.1. Построение графиков функций одной переменной

Система **MatLab** может создавать как плоские графики, так и трехмерные сетчатые поверхности, а также движущиеся графики, или анимацию.

В научных и технических задачах основными и наиболее часто употребительными являются графики, которые представляют собой кривые, описывающие те или иные численные данные. **MatLab** располагает набором команд высокого уровня, которые используются для построения таких кривых и управления ими. Это такие команды, как **plot, title, axis, text, hist, contour** и ряд других. Кроме того, строить графики, управлять ими и редактировать их можно с помощью инструментальных панелей графических окон. Можно также использовать комбинацию обоих подходов. Например, можно использовать команды для создания графиков, а затем модифицировать их.

Для того чтобы построить график функции **y=f(x)**, достаточно тем или иным способом сформировать два вектора одинаковой размерности - вектор значений аргументов **x** и вектор соответствующих значений функции **y**, а затем выполнить команду **plot**. Команда **plot** открывает графическое окно и отображает в нем зависимость **y(x)** в линейных осях, при условии что **x** и **y** являются векторами одинаковой длины.

Рассмотрим пример построения графика синуса на интервале от -4 до 4.

Command Window **Пример 3.5.3-52**

```
>>x = -4:.01:4;
>>y = sin(x);
>>plot(x,y);
>>
```

В результате выполнения команд открывается графическое окно, изображенное на рис. 3.5.3-8.

В вышеописанном примере вектор **x** является набором равноотстоящих точек с шагом **0.01**, а **y** - вектор со значениями функции синуса в этих точках. Для отображения графика система **MatLab** открывает отдельное окно с именем **Figure 1**. Переход между окнами, то есть возврат в окно **MatLab** или переход от одного графического окна к другому графическому окну осуществляется в соответствии с правилами среды: с помощью команд основного меню; комбинации **<Alt+Tab>** или с помощью мыши.

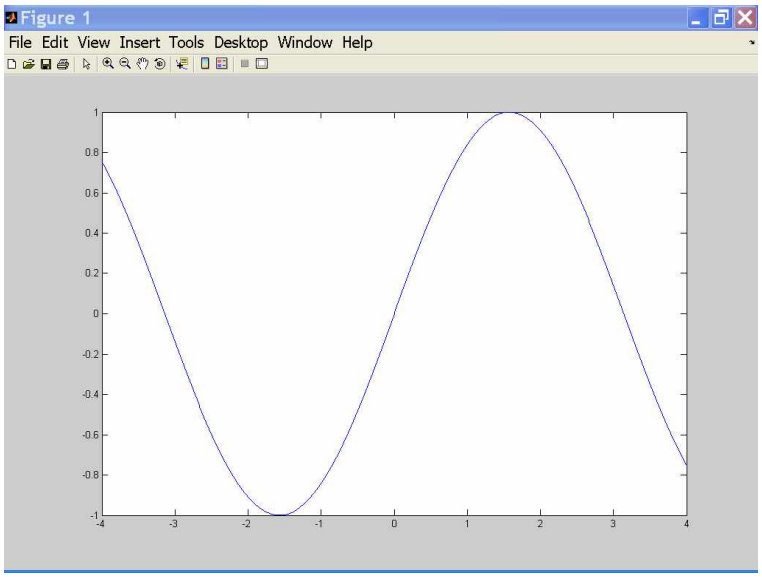


Рис. 3.5.3-8. Графическое окно с графиком функции **y=sin(x)**

В вышеописанном примере вектор **x** является набором равноотстоящих точек с шагом **0.01**, а **y** - вектор со значениями функции синуса в этих точках. Для отображения графика система **MatLab** открывает отдельное окно с именем **Figure 1**. Переход между окнами, то есть возврат в окно **MatLab** или переход от одного графического окна к другому

графическому окну осуществляется в соответствии с правилами среды: с помощью команд основного меню; комбинации **<Alt+Tab>** или с помощью мыши.

Рассмотрим еще один пример п**остроения графика $y=e^{-x2}$ на интервале [-1.5 ;1.5].**

Command Window **Пример 3.5.3-53**

```
>>  x = - 1.5:.01:1.5;
>>  y = exp(-x.^2);
>>  plot(x,y)
>>
```

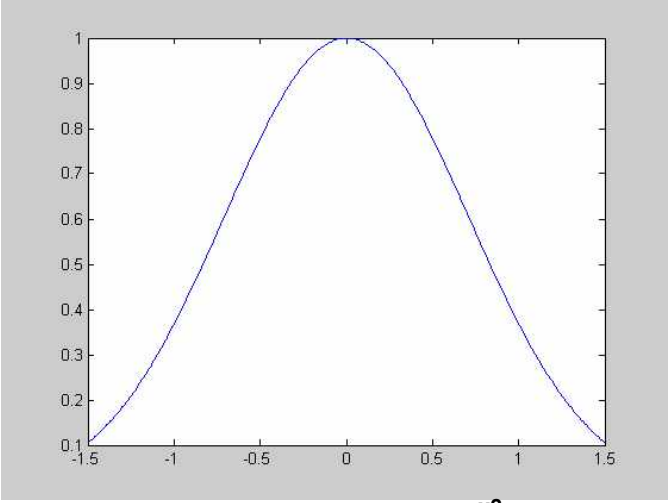


Рис. 3.5.3-9. График функции $y=e^{-x2}$

Необходимо обратить внимание на то, что точка перед знаком возведения в степень (**^**) обязательна, поскольку мы хотим, чтобы возведение в степень выполнялось поэлементно.

В системе **MatLab** можно также построить кривые, заданные параметрически.

Command Window **Пример 3.5.3-54**

```
>>  t=0:.001:2*pi;
>>  x=cos(3*t);
>>  y=sin(2*t);
>>  plot(x,y)
>>
```

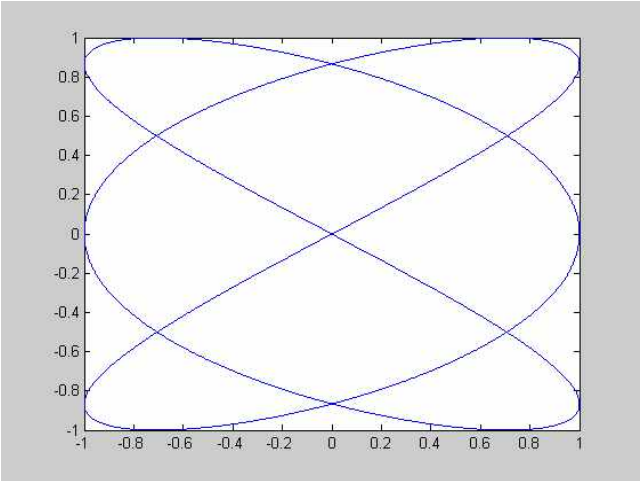


Рис. 3.5.3-10. График функции, заданный параметрически

Аргументами функции **plot()** могут быть различные комбинации векторов и матриц:

- **plot(y):**
  - если **y** - вектор, то будет нарисована кривая **y** как функция номера элемента в **y**;
  - если **y** - матрица, то будет сгенерирован набор кривых, каждая из которых представляет собой зависимость столбца матрицы от номера строки.
- **plot(x,y)**
  - если **x** и **y** - вектора одинаковой длины и размерности (оба строки или оба столбцы), то будет нарисована кривая **y** от **x**;
  - если **x** - вектор, а **y** - матрица, строки или столбцы **y** будут нарисованы в зависимости от **x**; если столбец матрицы **y** имеет ту же длину, что и вектор **x**, то будет построен набор кривых, представляющий зависимость каждого из столбцов от **x**; если строка матрицы **y** имеет ту же длину, что и вектор **x**, то будет построен набор кривых, представляющий зависимость каждой из строк от **x**; если число строк и столбцов **y** одинаково, то строятся столбцы от **x**;
  - если **x** -матрица, а **y** -вектор, то будет построено несколько кривых, представляющих зависимость **y** от строк или столбцов матрицы **x** по правилу, описанному в предыдущем пункте;
  - если **x** и **y** -матрицы одинаковой размерности, то будет построен набор кривых, представляющих столбцы **y** от столбцов **x**.

При вышеописанных способах вызова функции **plot()** различные кривые на одном и том же графике рисуются разным цветом. Перебор цветов выполняется автоматически, а при использовании соответствующих аргументов у команды **plot()** эти цвета можно выбирать.

В общем случае, число аргументов у команды **plot()** не ограничивается двумя, то есть **plot(x1,y1,x2,y2,...)**, причем правила, описанные выше, относятся к каждой паре аргументов.

С помощью соответствующих команд любой график в графическом окне **MatLab** может быть снабжен заголовком, именами осей, и на сам график может быть помещен дополнительный текст с помощью команд вывода текста. На график можно так же поместить сетку. Аргументами всех этих команд является текстовая строка. Например, команда **title('График наилучшего приближения')** добавит к вашему графику заголовок. Команда **gtext('Пятно')** позволяет с помощью мыши или клавишного курсора разместить на рисунке индикаторный крест, в месте размещения которого и будет помещен текст после нажатия произвольной клавиши. При необходимости сделать подписи осей используются команды **Xlabel('ПодписьX')**, **Ylabel('ПодписьY')**.

Для изображения нескольких кривых на одном рисунке существуют два способа, которые иллюстрируются следующими примерами.

Пример 3.5.3-55a

```
>> x=0:.01:2*pi;
>> y1=sin(x);
>> y2=sin(2*x);
>> y3=sin(4*x);
>> plot(x,y1,x,y2,x,y3)
>>
```

Command Window **Пример 3.5.3-55b**

```
>> x=0:.01:2*pi;
>> Y=[sin(x)', sin(2*x)', sin(4*x)'];
>>  plot(x,Y)
>>
```

Эти два примера полностью эквивалентны, а результат изображен на рис. 73.5.3-11. Отметим, что во втором примере формируется матрица **Y**, содержащая значения изображаемых функций в виде столбцов.

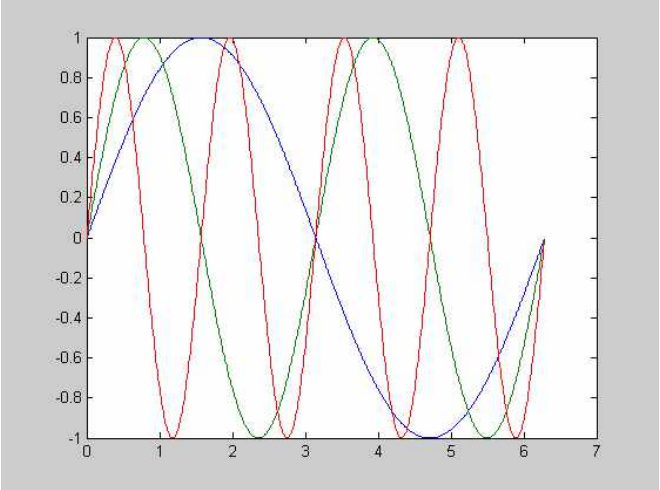


Рис. 3.5.3-11. Графики функций **y1=sin(x); y2=sin(2*x);y3=sin(4*x);**

Другим способом такого отображения нескольких функции на одних осях является использование команды **hold on**, которая замораживает текущий график, так что последующие кривые размещаются на этом же графике. При этом масштаб и разметка осей изменяются, если новая кривая не вписывается в нарисованные оси. Команда **hold off** приводит к тому, что любой последующий вызов команды **plot** создает новый рисунок в этом же окне, то есть предыдущий график стирается.

При выводе графика можно сменить принятый по умолчанию тип точек, с помощью которых рисуется данный график.

Command Window **Пример 3.5.3-56**

```
>> x=0:.01:2*pi;
>> y1=sin(x);
>> y2=sin(2*x);
>> y3=sin(4*x);
>> plot(x,y1,'--',x,y2,':',x,y3,'+')
>>
```

Результат работы программы приведет к тому, что первый график будет нарисован пунктиром, второй - точками, а третий - символами + (рис. 3.5.3-12). В общем случае каждая линия на графике определяется триплетом **x,y,s**, где **x** и **y** - это вектора с координатами функции, а **s** – строковая переменная, образованная любыми комбинациями из каких-нибудь приведенных ниже столбцов таблицы 3.5.3-8.

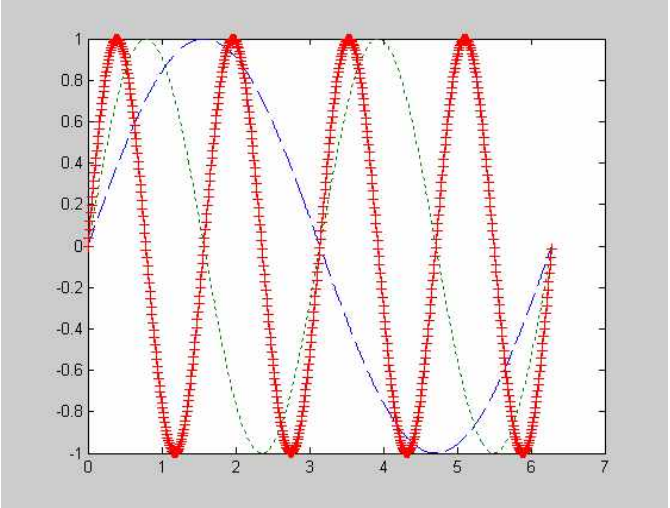


Рис. 3.5.3-12. Пример графика **Sin(x), Sin(2x) и Sin(3x)**

. Таблица 3.5.3-8

S	Цвет	S	Маркер
y	желтый	**.**	точка
m	фиолетовыйо	**кружок**	:
c	голубой	**x**	x-метка
r	красный	+	плюс
g	зеленый	*	звездочка
b	синий	**s**	квадрат
w	белый	**d**	алмаз
k	черный	**v**	Треугольник(вниз)
^	треугольник (вверх)	**-**	сплошная
<	треугольник (влево)		точечная
>	треугольник (вправо)	**-.**	Штрих - пунктирная
p	шестиугольник	**-**	пунктирная
h	восьмиугольник		

По умолчанию масштаб осей выбирается автоматически, таким образом, чтобы график целиком поместился в окне, причем с разумным запасом. Кроме того, оси автоматически размечаются, и по умолчанию выбирается, декартова система координат. Причем началом координат считается точка с координатами **xmin**, **ymin**, расположенная в левом нижнем углу. Эта автоматическая установка может быть изменена с помощью команды **axis**. Примеры использования этой команды применительно к плоским графикам описаны в табл. 3.5.3-9.

При необходимости вывести в одном графическом окне несколько графиков, то есть разбить графическое окно на несколько областей - подобластей, в каждой из которых отображаются свои оси, используется команда **subplot(m,n,p)**. Значение **m** указывает, на сколько областей окно разбивается по вертикали, **n** – по горизонтали, а **p** -порядковый номер подокна при счете слева направо и сверху вниз. Команда **subplot** используется как для создания нового подокна, так и для перехода от одного подокна к другому. После вызова этой команды **plot** построит график или графики в соответствующем подокне. Например, последовательность команд **Примера 3.5.4-57** строит два рисунка в верхней и нижней части экрана. В верхней части строится график **sin(x)**, а в нижней части экрана - зависимость **log(abs(sin(x)))**. Как видно из приведенной выше последовательности команд, при **x=0**

вычисляется **log(0)**. При этом в командном окне появляется **предупреждение**, а график строится в точках **−1 ≤ x ≤ 0.1** и **0.1 ≤ x ≤ 1**.

Таблица 3.5.3-9

Обращение	Результат
axis([xmin xmax ymin ymax])	Устанавливает пределы изменения по **x** и **y** координатам соответственно. Если максимальный предел по любой координате установить равным **Inf**, то соответствующий предел будет определяться автоматически. Аналогичное справедливо для нижнего предела, если его установить равным - **Inf**.
axis('square')	Делает область вывода квадратной.
axis('equal')	Делает единицы измерения по оси **x** и **y** одинаковыми.
axis('normal')	Восстанавливает режим по умолчанию.
axis('ij')	Помещает начало координат в левый верхний угол. Направление оси **x** - слева направо, направление оси **y** - сверху вниз.
axis('xy')	Восстанавливает стандартную декартову систему координат с началом в левом нижнем углу и направлением оси **y** снизу вверх.
axis('tight')	Устанавливает пределы по осям точно равными максимальным и минимальным значениям соответствующих переменных.
axis('off')	Делает невидимыми оси, метки осей и надписи
axis('on')	Включает оси и их разметку.

Command Window **Пример 3.5.3-57**

```
>> x= -1:.1:1;
>> y1=sin(x);
>> subplot(2, 1, 1), plot(x, y1);
>> y2=log(abs(y1));
>> subplot(2, 1, 2), plot(x,y2);
>>
```

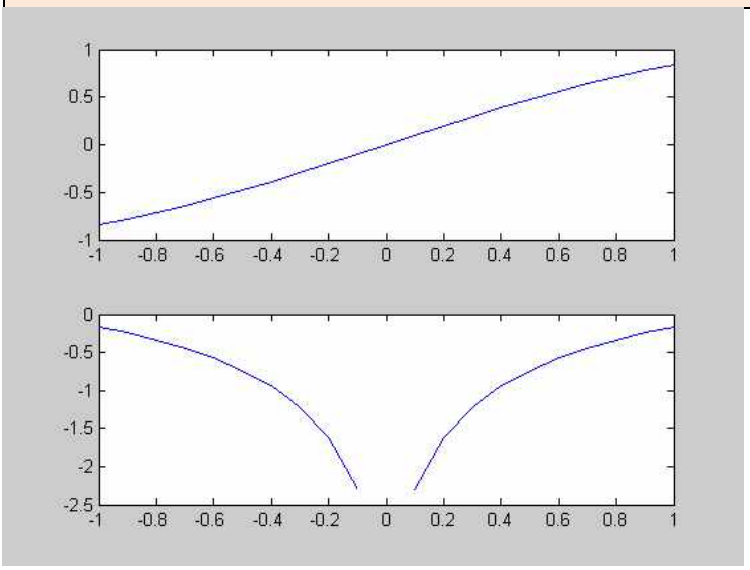


Рис. 3.5.3-13. Две подобласти в одном графическом окне

Полярными координатами (**ρ,θ**) точки **P** (рис. 3.5.3-14) называются длина **ρ** радиус - вектора, проведенного из начала координат в точку **P**, и угол **θ**, образованный между осью **x** и этим вектором. Угол отсчитывается от положительного направления оси **x** к радиус-вектору против часовой стрелки.

Довольно многие математические кривые с целью устранения неоднозначности принято задавать в полярной системе координат.

Для построения графиков функций, заданных в полярной системе координат, вместо функции **plot** используется функция **polar**:

Command Window **Пример 3.5.3-58**

```
>>  phi=0:0.1:2*pi;
>>  ro=5*(i+cos(phi));
>>  polar(phi,ro);
>>  title {'Кардиоида');
```

Результат построения кардиоиды для **a=5** приведен на рис. 3.5.3-14.

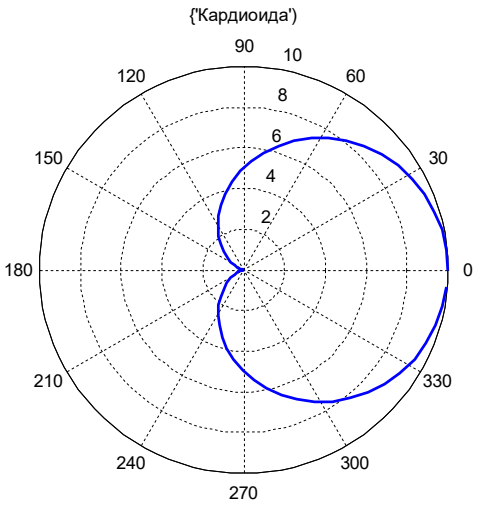


Рис. 3.5.3-14. График кардиоиды в полярной системе координат

Единицы измерения значений ординат и абсцисс далеко не всегда соответствуют друг другу, и для создания более обозримого графика вдоль одной или обеих координатных осей приходится выбирать логарифмический масштаб. Для построения таких графиков используются функции **loglog** (логарифмический масштаб по обеим осям), **semilogx** (логарифмический масштаб по оси **x**), **semilogy** (логарифмический масштаб по оси **y**).

Command Window **Пример 3.5.3-59**

```
>>  x=0:10:100;
>> y=exp(x);
>> semilogy(x,y)
```

Результат построения графика в логарифмическом масштабе приведен на рис. 3.5.3-15.

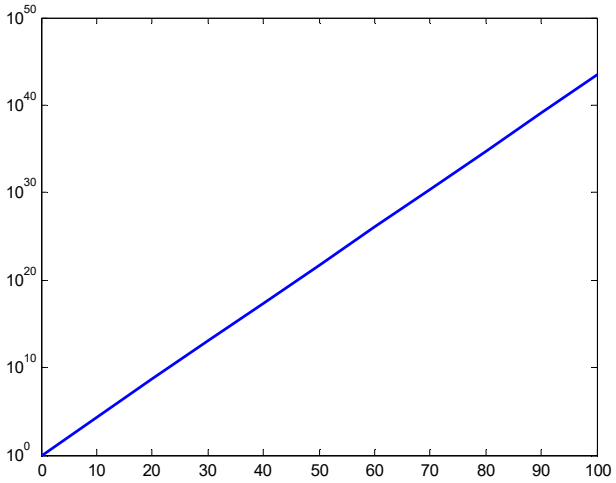


Рис. 3.5.3-15. График с логарифмическим масштабом по оси **y**

Довольно интересной модификацией функции **plot** является функция **fplot**, которая строит график функции **y=f(x)** без предварительного вычисления векторов (**x1,x2,…**) и (**y1,y2,...**). Базовый формат вызова этой функции включает два аргумента:

```
fplot (@name_fun, [limits])
fplot (' name_fun',   [limits])
```

В первом случае первым аргументом является указатель на функцию с именем **name_fun**, во втором случае - строка с именем обрабатываемой функции. Аргумент **limits** может быть представлен либо двухкомпонентным вектором **[xmin xmax]**, либо четырехкомпонентным вектором **[xmin xmax ymin ymax].** Укороченный вариант задает пределы изменения аргумента **x**, расширенный - дополнительно представляет пределы изменения функции. Например:

```
fplot(@sin,   [0 2*pi])
fplot('sin',   [0 2*pi])
```

По сравнению с **plot** функция **fplot** берет на себя вычисление таблицы значения функции, проявляя при этом некоторый интеллект - в местах резкого изменения функции значения аргумента **x** выбираются с более мелким шагом. Функция **fplot** гарантирует, что относительное уклонение воспроизводимой функции отличается от ее идеального графика не более чем на **0,2%.** Если вам нужен более точный или более грубый график, то после двух обязательных аргументов в функции **fplot** можно задать желаемую относительную погрешность - число, меньшее **1**:

```
fplot(@sin, [0 2*pi], 0.05)
```

В этом случае гарантировано построение графика, отличающегося от идеальной кривой не более чем на **5%**. По умолчанию точность равна $\mathbf{2 \cdot 10^{-3}}$, и она определяет, на сколько точек делить интервал, чтобы погрешность от линейной интерполяции не превосходила этой заданной точности.

Еще один дополнительный параметр — целое натуральное число **n** требует, чтобы функция **fplot** использовала при построении не менее чем **n+1** точку. Относительная погрешность и количество точек могут задаваться в любом порядке:

fplot(@sin, [0 2*pi], 0.05, 20)
fpiot(@sin, [0 2*pi], 20, 0.05)

Наконец, как и в функции **plot**, среди дополнительных параметров могут находиться строки, управляющие цветом и маркировкой графика. Последовательность задания всех дополнительных параметров – произвольная. Результат выполнения следующего оператора приведен на рис. 3.5.3-16.

Command Window **Пример 3.5.3-60**

```
>> fplot(@sin,   [0 2*pi],   0.05,   's--')
>>
```

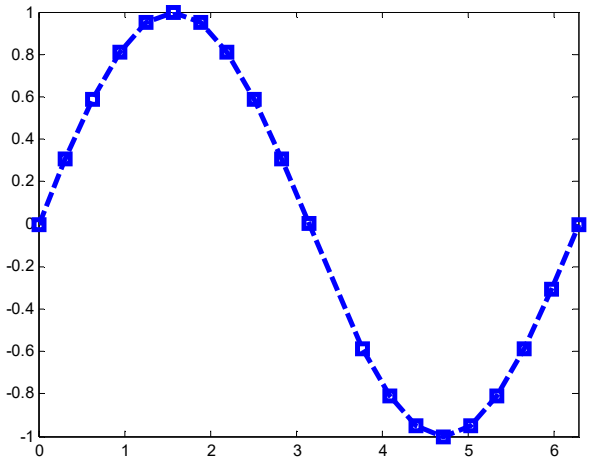


Рис. 3.5.3-16. Синусоида, построенная с помощью функции **fplot**

Функция **fplot** умеет возвращать значения компонентов векторов **x** и **y**, если к ней обратиться следующим образом:

[x y] = fplot(@sin, [0 2*pi]);

В этом случае кривая не рисуется, а соответствующие координаты заносятся в массивы **x** и **y** соответственно.

Функция **f**, указываемая в качестве первого аргумента **fplot**, кроме независимой переменной **x** может содержать дополнительные параметры, например, **y=f(x,a1,a2).** Тогда эти параметры указываются среди дополнительных аргументов после задания относительной погрешности и минимального числа точек:

fplot('nanie_fun', [xl x2], 0.05, 20, al, a2)

Рассмотрим пример 3.5.3-61, результат которого отображен на рис. 3.5.3-17.

Command Window **Пример 3.5.3-61**

```
>>  subplot (1, 2,1);
>>  fplot('[tan(x),sin(x),cos(x)]',2*pi*[-l 1 -1 1]);
>>  subplot (2, 2,2);
>>  fplot('sin(l / x)', [0.01 0.1);
>>
```

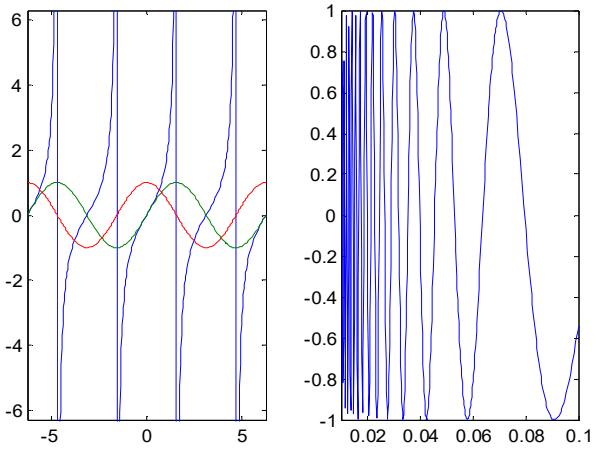


Рис. 3.5.3-17. Примеры работы функции **fplot**

Функция **fplot()** предоставляет альтернативную возможность изображения функций по сравнению с вычислением вектора **y** по **x** и последующим изображением этой кривой с помощью функции **plot()**. Эта функция бывает особенно полезной, когда кривая имеет несколько разных скоростей изменения и заранее не ясно, в скольких и каких точках необходимо вычислять и выводить кривую. Этой функции необходимо передавать строку, описывающую требуемую функцию в виде **f(x)**. Строка, описывающая **f(x)**, может содержать любые допустимые в **MatLab** операции и/или функции. Функция **f(x)** должна возвращать вектор той же размерности, что и **x**, или матрицу, каждый столбец которой имеет столько же элементов, сколько и **x**.

Например, для того чтобы нарисовать кривую **y = sin(x)•cos(2x)** в диапазоне **x** от **0** до **5π**, необходимо вызвать функцию **fplot ('sin(x) .*cos(2x)', [0 5*pi])**.

График с двумя осями ординат (одна ось отображается слева, другая ось справа) реализуется функцией **plotyy(x1,y1,x2,y2)** и той же функцией с добавлением параметров масштабирования **'f1'** или **'f1','f2'**, в роли которых могут выступать **plot, semilogx, semilogx, loglog** (рис. 3.5.3-18).

Command Window **Пример 3.5.3-62**

```
>>  x=0:0.01:12*pi;
>>  plotyy(x,sin(x).*exp(-0.1.*x),x, 10*exp(-0.1.*x)) %
```

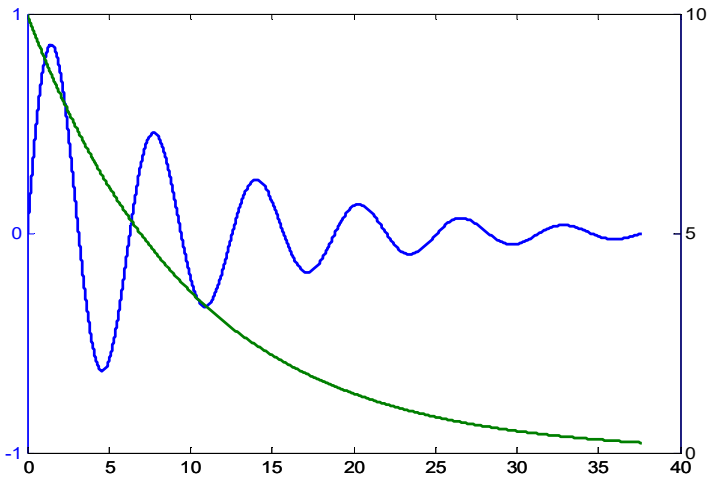


Рис. 3.5.3-18. Графическое окно с двумя осями координат

Для размещения различных надписей на полученном рисунке применяют функции **xlabel, ylabel, title** и **text.** Функция **xlabel** предназначена для размещения горизонтальной оси, функция **ylabel** – названия вертикальной оси. Для размещения надписи в произвольном месте рисунка используется функция **text**, а общий заголовок для графика выводится функцией **title.** Кроме того, используя команду **grid on** можно нанести измерительную сетку.

Рассмотрим пример 3.5.3-62, результат которого приведен на рис. 3.5.3-19.

Command Window **Пример 3.5.3-63**

```
>>  x=1.5 : 0.1 : 2.5;
>>  y=sin(x);
>>  title ('График sin(x)');
>>  xlabel('ось x');
>>  ylabel('sin(x)');
>>  text(2.1,0.9,f(x)=sin(x)');
>>grid  on
>>plot(x, y)
>>
```

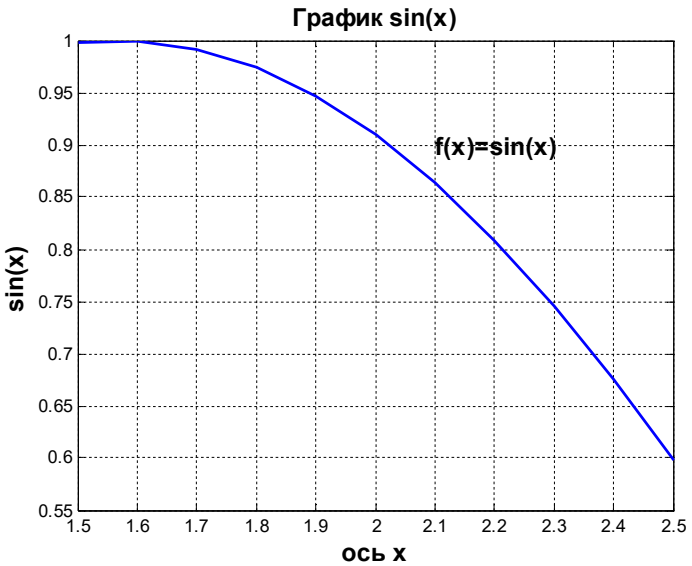


Рис. 3.5.3-19

В системе **MatLab** существует функция, которая создает ступенчатое изображение ваших данных. Например, вместо соединения каждой пары точек из векторов аргументов прямыми линиями (как это делает функция **plot()** или **line()**) функция **stairs()** изображает ваши данные в виде горизонтальных отрезков на уровне **yi,** причем каждый отрезок горизонтальной линии длится от **i** до **i + 1**, если обращение имеет вид **stairs(y)**, и от **xi** до **xi+1**, если обращение имеет вид **stairs(x,y).** Значения **xi** не должны быть равноотстоящими и не должны быть упорядочены по возрастанию. Вывод графика на экран можно предотвратить, используя обращение вида **[xs,ys]=stairs(x,y).** Потом этот график может быть выведен на экран с помощью команды **plot(), line()** или другим каким-нибудь способом.

3.5.3.2.2. Инструментальная панель графических окон

В предыдущих разделах для отображения, оформления и исследования графиков функций использовались специальные команды системы **MatLab,** которые вводились с клавиатуры в командном окне. Однако большую часть выполненных действий можно сделать также с помощью команд инструментальной панели графических окон.

На рисунке 3.5.3-20 показаны базовые компоненты типичного графического окна.

Рис. 3.5.3-20. Компоненты графического окна

Команды основного меню графического окна представляют большой набор средств для отображения и оформления графиков, позволяющих в интерактивном режиме и без использования команд, придать графику желаемый вид, а также снабдить его аннотациями и дополнительными элементами. Что касается кнопок инструментальной панели, то они дублируют наиболее используемые пункты меню, ускоряя к ним доступ.

В полной мере научиться быстро и эффективно пользоваться инструментальными средствами графических окон системы **MatLab** можно только на практике. В этом разделе остановимся лишь на некоторых возможностях построения и управления графиками с помощью инструментальных средств.

Из приводившихся выше примеров видно, что графики функций отображаются в специальном окне, снабженном меню и панелью инструментов. Это окно соответствует всем стандартам **Windows** - его можно перемещать по экрану, изменять размеры, сворачивать и разворачивать, удалять. В соответствии с объектно-ориентированной терминологией **MatLab,** оно представляет объект типа **Figure** (графическое окно).

Меню, встроенное в графическое окно, позволяет:

- обратиться к довольно мощному редактору свойств графических объектов;
- сохранить графическое окно в файле с расширением **MatLab**;
- перейти в среду **MatLab** и выполнить любые действия в командном окне.

На рис. 3.5.3-21 приведены команды наиболее важных элементов меню графического окна.

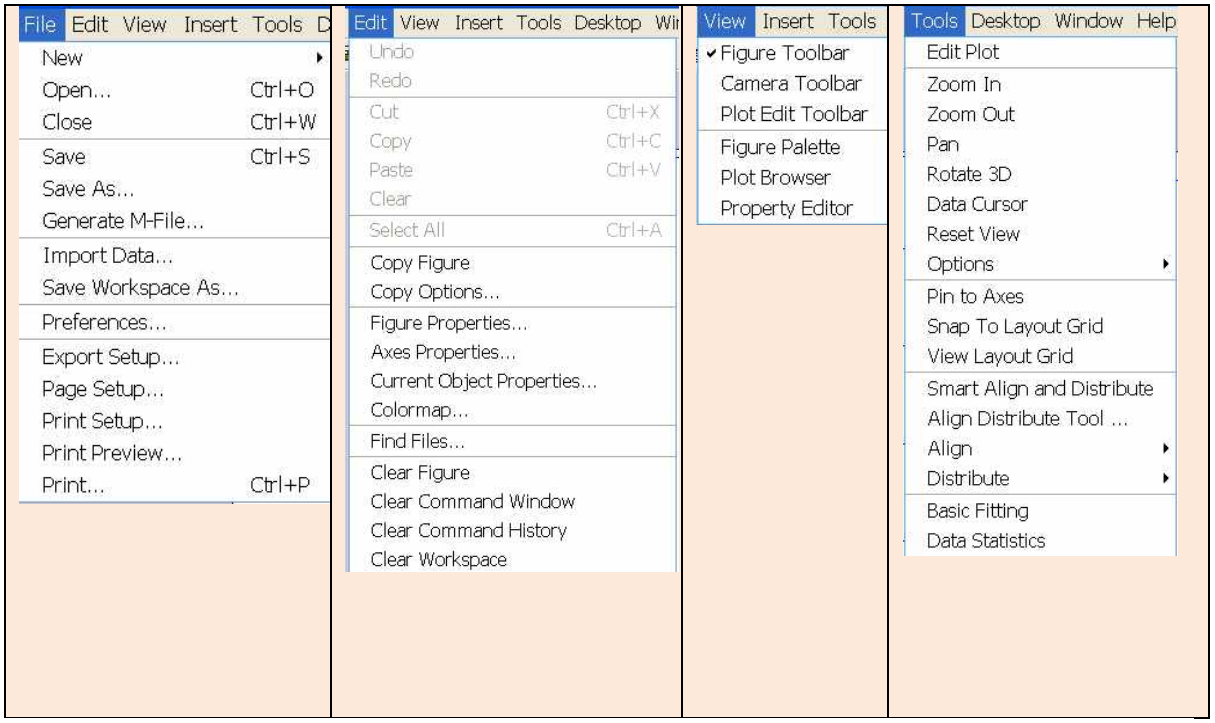


Рис. 3.5.3-21. Команды элементов меню графического окна

Первая группа команд элемента меню **File** (Файл) позволяет закрыть графическое окно (**Close** (Закрыть)), вызвать в него ранее сохраненный графический файл (**Open** (Открыть)) или начать вручную создавать новый график, используя при этом возможности **Property Editor** (Редактор свойств) графического объекта (**New Figure** (Новый графический объект)).

Команды второй группы позволяют сохранить графическое окно под прежним (**Save** (Сохранить)) или новым (**Save As** (Сохранить как)) именем, а также экспортировать его в нужном вам графическом формате (**Export Setup** (Параметры экспорта)).

Очень интересен результат выполнения команды **Generate M-File** (Генерировать **m-файл**). Она позволяет автоматически создать функцию построения графиков, расположенных в графическом окне. Отметим, что к такого рода процедуре следует прибегать только в том случае, когда пользователь провел вручную достаточно большую редакторскую правку.

Команды **Import Data** (Импортировать данные) и **Save Workspace As** (Сохранить рабочее пространство как) уже упоминались при описании главного меню.

Команда **Preferences** (Предпочтения) выводит нас в окно установок параметров системы **MatLab**. Аналогичная возможность входит в состав меню **File** (Файл) командного окна.

Команда **Export Setup** (Параметры экспорта) вызывает диалоговое окно, в котором можно просмотреть и изменить значения текущих параметров графического окна и графика — размеры **(Size**), способ визуализации (**Rendering**), характеристики шрифта (**Font**) и свойства линий (**Lines**). После этого откорректированный график можно сохранить в файле с расширением **fig.**

Оставшиеся команды последней группы меню **File** (Файл) связаны с подготовительными работами по настройке параметров графика, листа бумаги и принтера перед выдачей окончательной команды вывода (**Print** (Печать)).

Элемент меню **Edit** (Правка) - включает набор стандартных команд, присутствующих в любом текстовом или графическом редакторе — **Undo** (Отменить ввод),

Cut (Вырезать), **Copy** (Копировать), **Paste** (Вставить), **Clear** (Очистить) и **Select All** (Выделить все).

Команда **Figure Properties** (Свойства графического объекта) вызывает **Property Editor** (Редактор свойств), который пристыковывается к графическому окну) и предварительно настроен на изменение свойств фигуры (**Figure**). В появившихся окнах редактора отображаются наиболее употребительные параметры фигуры. Вы можете изменить заголовок графического объекта (**Figure Name**), сменить цветовую палитру (**CoJormap**); указать цвет окаймления поля графика (**Figure Color**), включить или отключить воспроизведение порядкового номера графического окна (**Show Figure Number**). С помощью кнопки **Inspector** (Инспектор) вызывается **Property Inspector** (Инспектор свойств), в котором отображаются все свойства фигуры. Любое из них, не включенное в наиболее употребительный набор, тоже может быть изменено.

Если в графическом окне выделить оси, щелкнув по одной из них, то в окнах **Property Editor** (Редактор свойств) появляются свойства осей и меняется заголовок (**Property Editor — Axes)**. Вы можете изменить заголовок, расположенный над полем графика (**Title**), установить цвета осей, подписей и фона поля графика (**Colors**), включить или отключить изображение координатной сетки по любой из осей (**Grid**), включить или отключить обводку рамки поля графика (**Box**). Для каждой из координатных осей можно изменить наименование оси **(X Label**, **Y Label, Z Label)**, минимальный и максимальный пределы изменения соответствующей переменной (**X Limits, Y Limits, Z Limits**), способ масштабирования по каждой из осей (**X Scale, Y Scale, Z Scale**), сменить направление оси на противоположное (**Reverse**). Кнопка **Inspector** (Инспектор) в окне **Property Editor** (Редактор свойств) вызывает инспектор, отображающий значения всех свойств координатных осей.

Выделение той или иной кривой приводит к перенастройке редактора свойств на параметры указанного графика (**Property Editor — Line-series**). Раскрывающиеся списки **X/Y/Z Label Data Source** (Источник данных) позволяют изменить значения векторов **x, y** и **z,** задающих табличную функцию. После изменения любого из этих элементов необходимо нажать кнопку **Refresh Data** (Обновить данные), и тогда новые значения вступят в силу.

В раскрывающемся списке **Plot Type** (Тип графика) вы можете выбрать один из пяти способов отображения графика функции. По умолчанию график воспроизводится линией (Line, отсюда и фрагмент заголовка окна — **Lineseries**). Четыре других способа воспроизведены на рис. 3.5.3-21. В раскрывающемся списке **Line** (Линия) задается тип линии (сплошная, штриховая, пунктирная или штрих-пунктирная). Значение **no line** (нет) заменяет линию графика маркерами в заданных точках. Цвет линии и ее толщина, регулируются значениями, установленными при помощи кнопок-списков, выделяемых подсказками **Color** (Цвет) и **LineWidth** (Толщина линии) соответственно. В списке **Marker** можно выбрать подходящую конфигурацию маркера, размер которого задается значением в раскрывающемся списке **MarkerSize.** Цвета контура маркера и его внутренней области устанавливаются с помощью кнопок-списков **MarkerEdgeColor** и **MarkerFace-Color** соответственно.

Команды **Axes Properties** (Свойства осей) и **Current Object Properties** (Свойства объекта) вызывают **Property Editor** (Редактор свойств) с одновременной настройкой на параметры соответствующего объекта.

Команды последней группы обеспечивают очистку поля фигуры, командного окна, окна истории команд и рабочего пространства. По умолчанию очистка трех последних компонентов сопровождается запросом подтверждения на выполнение операции.

Кроме кнопок 1, 2, 3, 4 панели инструментов (рис. 3.5.3-20), обеспечивающих создание, открытие, сохранение и печать графического объекта, на панели инструментов находятся следующие кнопки (рис. 3.5.3-20):

5 Edit Plot — включение режима выделения графического объекта;

6 Zoom In — увеличение масштаба изображения;

7 Zoom Out — уменьшение масштаба изображения;

10 Data Cursor — включение режима, при котором точка графика, попавшая под курсор, сопровождается отображением значений координат;

11 Insert Colorbar — включение столбика с палитрой цветов (**Colormap**), отражающего распределение высот **z**;

12 Insert Legend — вставка автоматической легенды;

13 Hide Plot Tools — удаление средств редактирования и просмотра;

14 Show Plot Tools — пристыковка к графическому окну средств редактирования и просмотра (**Figure Palette** (Редактор графического окна), **Property Editor** (Редактор свойств), **Plot frowsier** (Просмотр графика)).

Команды элемента меню **View** (Вид) позволяют отобразить (команда помечена галочкой) на экране или скрыть следующие средства редактирования (рис. 3.5.3-21):

Figure Toolbar — панель инструментов графического окна;

Camera Toolbar — панель инструментов для управления камерой;

Plot Editor Toolbar — панель инструментов для редактирования графика;

Figure Palette — редактор графического окна;

Plot Browser — средства просмотра графических объектов;

Property Editor — редактор свойств графических объектов.

На рис. 3.5.3-22 представлено графическое окно со средствами редактирования.

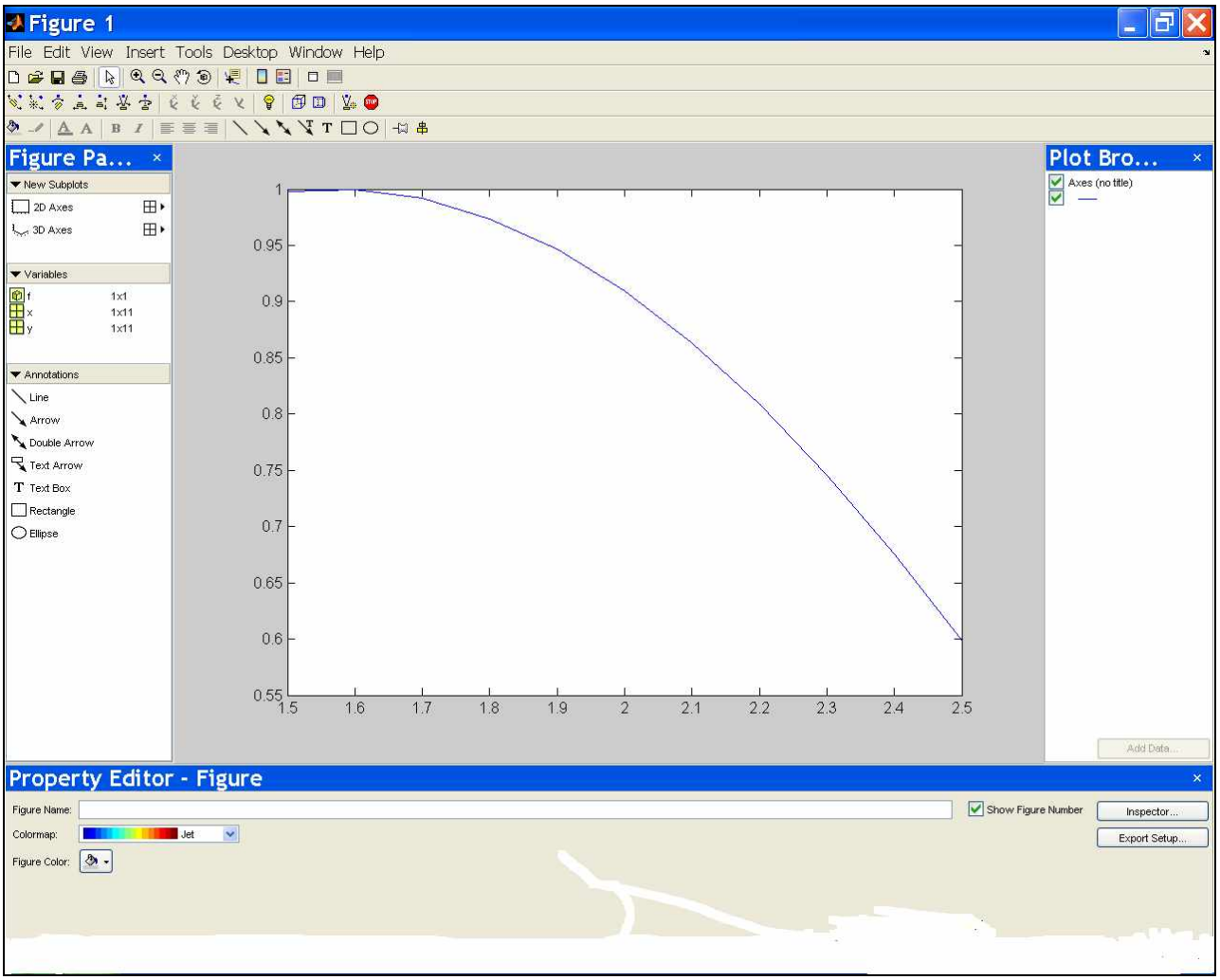


Рис. 3.5.3-22. Графическое окно со средствами редактирования

Окно **Plot Browser** (Просмотр графика) отображает список графических объектов, расположенных в поле графика.

Элемент меню **Insert** (Вставка) содержит набор команд по вставке в графическое окно различных элементов оформления — легенды (**Legend**), разноцветной полоски (**Colorbar**), линии (**Line**), различных стрелок (**Arrow, Double Arrow, Text Arrow**), пояснительной подписи (**Text Box**), прямоугольников (**Rectangle**) и эллипсов (**Ellipse**). Все эти команды дублируют возможности **Figure Palette** (Редактор графического окна) и **Plot Edit Toolbar** (Панель инструментов редактирования графика). Команда **Insert | Light** (Вставка | Свет) подключает к графическому окну **Property Editor** (Редактор свойств) источник света (**Property Editor — light**), с помощью которого можно изменить позицию (**Position — x, y, z**), стиль (**Style**) распространения лучей света (бесконечно удаленный — **Infinite**, локальный — **Local)** и цвет лучей (**Color**).

Элемент меню **Tools** (Сервис) содержит команды **Edit Plot** (Редактировать график), **Zoom In** (Увеличить масштаб). **Zoom Out** (Уменьшить масштаб), **Pan** (Захватить), **Rotate 3D** (3D-поворот) и **Data Cursor** (Данные под курсором) продублированы аналогичными кнопками на панели инструментов **Figure Toolbar** (Панель графического объекта). Команда **Reset View** (Восстановить вид) возвращает на исходное место график, перемешенный в режиме **Pan** (Захватить). Команда **Options** (Параметры) открывает список команд, с помощью которых можно изменить режим масштабирования, траекторию перемещения графика, захваченного в режиме **Pan** (Захватить), и способ отображения координат точки, выделенной курсором.

Команда **Tools | Data Statistics** (Сервис | Статистика данных) позволяет получить стандартные статистические данные о кривой, представленной в поле графика.

Довольно много средств оформления графиков представлено в окне **Figure Palette** (Редактор графического окна) и продублировано соответствующими кнопками на панели инструментов **Plot Edit Toolbar** (Панель инструментов редактирования графика

3.5.3.2.3. Построение трехмерных изображений

Не всегда весь объем визуализируемой информации удобно представить в виде набора двумерных кривых. Иногда, по смыслу задачи, данные должны быть представлены как объекты трехмерного пространства. Для изображения таких объектов рассмотрим несколько функций.

Способ обращения к команде **plot3**, а также список дополнительных параметров полностью совпадает с описанным в п. 3.5.3.3.1 и далее. Аналогично функциям **plot( ) - plot3( )** существует пара функция **comet( ) - comet3( )**. Функция **comet3( )** используется так же, как и **comet**. При использовании для обрисовки трехмерной кривой функции (объекта) **line( )** можно обращаться к этой функции как к функции **plot( )**, т.е. **line(x,y,z)**.

Трехмерные поверхности обычно описываются функцией двух переменных **z(x, y).** Специфика построения трехмерных графиков ("по точкам") требует не просто задания ряда значений векторов **x** и **y**. Она требует определения для **X** и **Y** двумерных массивов — матриц, которые используются для дальнейших вычислений и построений трехмерных поверхностей. Для создания таких массивов служит функция **meshgrid( )**. В основном она используется совместно с функциями построения графиков трехмерных поверхностей.

Функция **meshgrid( )** записывается в следующих формах:

[X,Y] = meshgrid(x) — аналогична [X,Y] = meshgrid(x,y);

[X,Y,Z] = meshgrid(x,y,z)

и возвращает трехмерные массивы, используемые для вычисления функций трех переменных и построения трехмерных графиков;

[X.Y] = meshgrid(x.y)

преобразует область, заданную векторами **x** и **y**, в массивы **X** и **Y**, которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков. Строки выходного массива **X** являются копиями вектора X.

Command Window **Пример 3.5.3-64**

```
>> [X,Y]=meshgrid(0:3,-3:0);
>>  X =
     0   1   2   3
     0   1   2   3
     0   1   2   3
     0   1   2   3
>>  Y =
    -3  -3  -3  -3
    -2  -2  -2  -2
    -1  -1  -1  -1
     0   0   0   0
```

Эти данные используются функциями

mesh(X,Y,Z) – построение сетчатого графика, где **X,Y,Z** – матрицы одинаковой размерности;

plot3(X,Y,Z) – построение точек, соединенных отрезками прямых (аналог **plot);**

urf(X,Y,Z) –построение поверхности и др.

Кроме численных методов построения можно использовать символьные: **ezmesh( ), ezsurf( ), ezplot3( )** и т. п. Синтаксис использования этих функций прост.

Рассмотрим пример 3.5.3-64 построения графика синуса на интервале от -4 до 4.

Command Window **Пример 3.5.3-64**

```
>>ezsurf('sin((x^2+y^2)^(0.5))/((x^2+y^2)^(0.5))',[-10 10],[-10 10])
>>
```

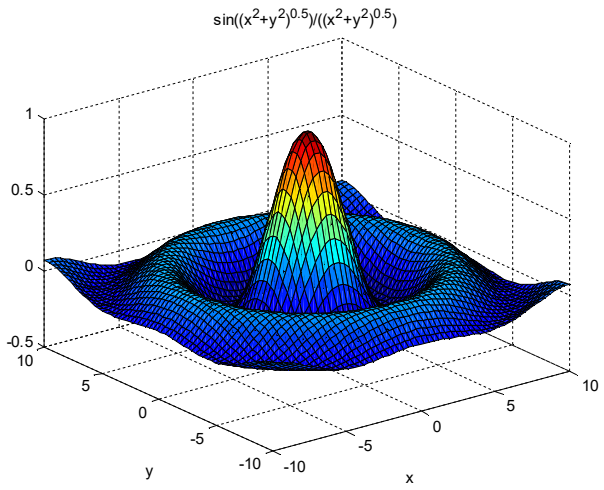


Рис. 3.5.3-23

Заключенное в апострофы – это функция, график которой необходимо построить, в скобках указаны пределы (их можно и не задавать). Обратите внимание на отсутствие . и **eps** – этого просто не надо.

Кроме этого, эти функции позволяют строить графики, заданные параметрически, задавать различные параметры графиков и другие возможности (посмотрите **help**).

Трехмерные сеточные поверхности изображаются с помощью функции **mesh( ).** Команда **mesh(z)** изображает в трехмерной перспективе поверхность, описываемую элементами матрицы **z**. Эта поверхность определяется **Z**-координатами точек над прямоугольной сеткой в **X-Y** плоскости. Попробуйте выполнить самостоятельно команду **mesh(eye(10))**. Для того чтобы нарисовать функцию **z=f(x,y)** над прямоугольником, необходимо определить вектора **xx** и **yy** , которые определят разбиение сторон прямоугольника. С помощью команды **meshgrid** можно создать матрицу **x**, каждая строка которой будет совпадать с xx, а размер столбцов которой будет совпадать с длиной вектора **yy**, и, аналогично, матрицу **y**, каждый столбец которой совпадает с **yy**, следующим образом **[x,y] = meshgrid(xx,yy);.** После этого необходимо определить матрицу **z**, вычисляя каждый ее элемент как функцию **f** в соответствующих точках, определяемых матрицами **x** и **y**, после чего использовать команду **mesh.** Вы можете, например, нарисовать поверхность на квадрате **[-2,2] x [-2,2]** с помощью следующего набора команд

Command Window **Пример 3.5.3-65**

```
>>xx = -2:.1:2;
>>yy = xx;
>>[x,y] = meshgrid(xx,yy);
>>mesh(z)
>>
```

Первые три строки можно заменить на **[x,y] = meshgrid(-2:.1:2, -2:.1:2)**; .

Более полно с возможностями изображения трехмерных поверхностей можно познакомиться с помощью оперативной помощи **(help plot3/mesh/surf).**

Одним из популярных способов визуализации поверхностей является изображение изолиний. **MatLab** предоставляет возможность построения изолиний двух типов - двумерных или плоских, фактически являющихся проекциями соответствующих линий постоянного значения на плоскость **X-Y**, и трехмерных изолиний, нарисованных в какой-либо перспективе.

Простейшим способом изобразить изолинии на плоскости -это обратиться к функции **contour(Z)**, где **Z** -это матрица, содержащая значения исследуемой поверхности на равномерной сетке, т.е. **Zij = f(i, j).** При таком обращении система **MatLab** сама выберет число изолиний и значения функции, при которой они будут построены. Если вы хотите сами задать число выводимых изолиний, то необходимо обращение вида **contour(Z,n_of_lines),** если же необходимо задать сами эти уровни, то это можно сделать с помощью обращения **contour(Z,values_of _levels)**. Если вам необходимо нарисовать одну изолинию с определенным значением функции, то вектор, содержащий уровни, должен иметь два одинаковых элемента, равных этому уровню.

Отмеченные выше три способа вызова функции построения изолиний строят их по отношению к номерам строк и столбцов матрицы **Z** так, что элемент **Z(1,1)** будет помещаться в левом нижнем углу рисунка. Можно построить изолинии относительно выбранных вами масштабов по оси **X** и **Y** . Для этого необходимо передать функции вектора

или матрицу координат, определяющие координаты каждого из элементов матрицы **Z**. Эта передача осуществляется путем одного из следующих обращений:

Command Window **Пример 3.5.3-66**

```
>>contour(x,y,Z); contour(x,y,Z,n_of_lines);
>>contour(x,y,Z,values_of_levels);
>>
```

В трехмерной графике выполняются представления функции **z=z(x,y),** отличающиеся способом соединения точек: линия, сечения, сетчатая или сплошная поверхность.

plot3(x,y,z) в тех же вариациях, что и **plot( ),** предполагает задание одномерных и двумерных массивов (строятся точки с координатами **x(i,:),y(i,:),z(i,:)** для каждого столбца и соединяются прямыми линиями. Если используется **[x,y]=meshgrid( ),** то строятся сечения.

Command Window **Пример 3.5.3-67**

```
>> t=0:pi/50:10*pi;
>>[x,y]=meshgrid([-2:0.1:2],[-2:0.01:2]);
>> plot3(sin(t),cos(t),t) %
>>z=exp(-x.^2-y.^2);
>>plot3(x,y,z)
>>
```

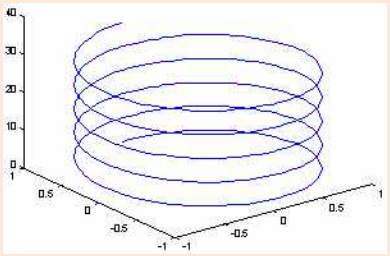


Рис. 3.5.3-24

В примере **mesh(x,y,z,c), mesh(z,c), mesh(z)** определяют задание сетчатой поверхности (массив с определяет цвета узлов поверхности; если **x,y** не указаны, то **x=1:n**, **y=1:m,** где **[m,n]=size(z).**

Command Window **Пример 3.5.3-68**

```
>> [x,y]=meshgrid(-8:0.5:8);
>> t=sqrt(x.^2+y.^2)+0.001;
>> z=sin(t)./t;
>> mesh(x,y,z) % Рис.9.8
>> meshc(x,y,z) %Рис.9.9
>>
```

Аналогичная функция meshc( ) в дополнение к поверхности строит проекции линий уровня, а meshz( ) делает срез поверхности до нулевого уровня (своеобразный пьедестал).

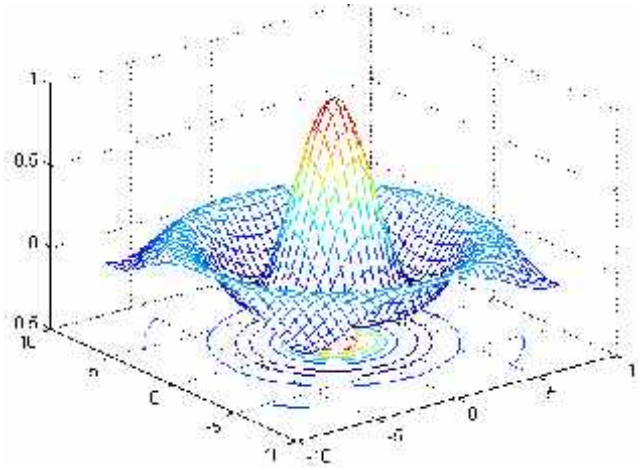


Рис. 3.5.3-25

Функции **surf(x,y,z,c), surf(z,c), surf(z)** определяют задание сплошной поверхности, отличаясь от mesh( ) системой окраски; аналогичная функция **surfc( )** задает проекции линий уровня.

Реализация трехмерной графики может сопровождаться множеством вспомогательных команд, например:

hidden on/off включает или выключает режим удаления невидимых линий (по умолчанию on);

shading faceted / flat / interp устанавливает затенение поверхностей (по умолчанию **faceted** дает равномерную окраску ячеек с черными гранями, **flat** - цветами узлов сетки, **interp** - интерполяцией цветов.

Создание графического объекта исходит автоматически при обращении к командам, порождающим объекты **Line( )** и **Surface( ),** но может выполняться и командой

axec('<имя свойства>',<значение>, ...)

Есть и команды более высокого уровня:

axis([xmin xmax ymin ymax]), axis([xmin xmax ymin ymax zmin zmax]), которые

устанавливает масштаб по осям;

axes off/on

выключает (включает) вывод на координатные оси обозначений и маркеров;

grid on/off, grid

ключает (выключает) или переключает режим нанесения координатной сетки на осях;

box on/off, box

включает (выключает) или переключает режим рисования контура параллелепипеда, трехмерный объект;

zoom on/off

включает (выключает) режим интерактивного масштабирования графиков (левая мышь около точки увеличивает масштаб вдвое, правая - уменьшает; удержанием левой мыши можно выделить прямоугольную область для детального просмотра; zoom out восстанавливает исходный график.

В отличие от meshc (...) и surfc(...) функция contour рисует только линии уровня соответствующих поверхностей и выступает в многообразии синтаксических форм: contour(X,Y,Z) - для массива Z =Z(X,Y), contour(X,Y,Z,n) - то же с указанием числа линий уровня (по умолчанию 10), contour(X,Y,Z,v) - то же для массива указанных значений ; contour(Z), contour(Z,n), contour(Z,v) - аналогичные функции без указания диапазонов для аргументов и contour(...,LineSpec) - аналогичные функции с указанием типа и цвета линий (см. plot); [C,h]=contour (...) возвращает массив C и вектор дескрипторов, позволяя тем самым продолжить работу с рисунком (давать оцифровку линий, заголовки и др.).

Функция contourf(...) закрашивает области между линиями уровня, аналогична contourf(...) с разницей в формате[C,h, cf]=contour (...), где cf определяет матрицу раскраски.

Пример 3.5.3-69

```
>> [x,y]=meshgrid(-8:0.5:8);
>> t=sqrt(x.^2+y.^2)+0.001;
>> z=sin(t).^3./t;
>> [c,h]=contour(x,y,z,20);
>>
```

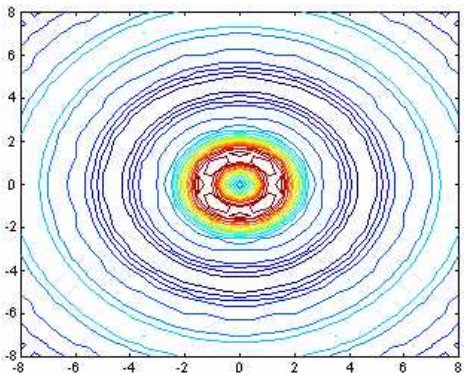


Рис. 3.5.3.26

Пример 3.5.3-70

```
>> [x,y]=meshgrid(-2:0.25:2);
>> t=sqrt(x.^2+y.^2)+0.001;
>> z=sin(t).^3./t;
>> [c,h,cf]=contourf(x,y,z,4);
>>
```

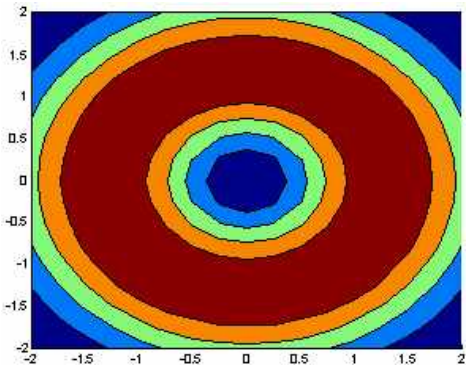


Рис. 3.5.3.27

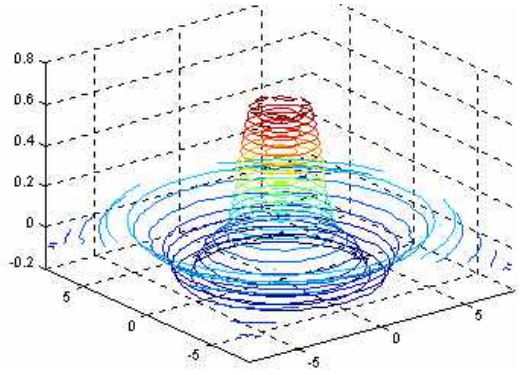


Рис. 3.5.3.28

Функция contour3(...) по синтаксису полностью аналогична contour(...), но изображает не проекции линий уровня, а рисует их в пространственной интерпретации; так команда [c,h]=contour3(x,y,z,20); дает фигуру (рис. 3.5.3.28).

3.5.3.3. Символьные (аналитические) вычисления и алгебраические преобразования

3.5.3.3.1 Введение в символьные вычисления

Система **MatLab** является самой крупной системой компьютерной математики, ориентированной на матричные и численные вычисления. Однако **MatLab** имеет также и средства аналитических вычислений. Пакет **Symbolic Math Toolbox** добавил системе **MatLab** качественно новые возможности, связанные с выполнением символьных вычислений и преобразований, которые были доступны только в системахпринципиально иного класса, относящихся к компьютерной алгебре. Теперь **MatLab**, с учетом новых средств, становится в полной мере универсальной системой. Последняя реализация системы символьной математики **Maple** в своем ядре и в расширениях имеет около 3000 функций. Система **MatLab** с пакетом **Symbolic**, включающим в себя чуть больше сотни символьных команд и функций, намного уступает **Maple** по количеству таких команд и функций. Кроме того, есть специальная команда, которая дает доступ к ядру **Maple,** что заметно расширяет круг используемых функций.

С помощью команды **help symbolic** можно получить перечень входящих в пакет команд и функций. Для получения справки по любой команде или функции можно использовать команду **help sym / name.m**, где **name** — это имя соответствующей команды или функции, а **name.m** — имя **m-файла**, задающего данную команду или функцию. Пакет **Symbolic Math Toolbox** включает следующие типы математических вычислений, которые приведены в таблице 3.5.3-10.

Таблица 3.5.3-10

Вычисления	Дифференцирование, интегрирование, пределы, суммы и произведения, разложение в ряд Тейлора
Линейная алгебра	

3.5.3.3.2 Создание и работа с символьными переменными, выражениями и функциями

Поскольку переменные системы **MatLab** по умолчанию не определены и традиционно задаются как векторные, матричные, числовые и т. д., то есть не имеющие отношения к символьной математике, для реализации символьных вычислений нужно, прежде всего позаботиться о создании специальных **символьных переменных**. В простейшем случае их можно определить как строковые переменные, заключив имена в апострофы. Например,

Command Window **Пример 3.5.3-71**

```
>> sin(x)^2 + cos(x)^2
??? Undefined function or variable 'x'.
>> sin('x')^2 + cos('x')^2
ans =  1
>>
```

Мы видим, что система **MatLab** «возмутилась» нашей небрежностью и сообщила, что функция или переменная **x** не определена и ни о каких вычислениях синуса и косинуса речи быть не может. Вместе с тем она подсказала, как надо поступить - заключить имя переменной в апострофы, ибо таким образом система получает информацию о необходимости включить символьный режим вычислений. Поэтому во второй раз получен вполне осмысленный результат - сумма квадратов синуса и косинуса переменной **'x'** выдана равной **1**.

Таким образом для работы с командами ядра **Maple** в **MatLab** необходимо создать новый символьный объект, который фактически является строковой переменной. То есть для проведения аналитических (символьных) операций нужно, чтобы соответствующие переменные были предварительно объявлены с помощью функции **sym():**

имя_переменной = sym('имя_переменной')

Например,

Command Window **Пример 3.5.3-72**

```
>>x = sym ('x')
x=   x
>>a=sym ('alfa')
A    alfa
>>
```

Рассмотрим пример, иллюстрирующий различие между стандартными типами данных **MatLab** символьных объектов.

Command Window **Пример 3.5.3-73**

```
>>sqrt(2)
ans=    1.4142
>>a=sqrt(sym(2))
a=      2^(1/2)
>>double(a)
ans=    1.4142
>>
```

Пакет **Symbolic** может работать с числами различных форматов, применяемых в **MatLab.** Рассмотрим следующий пример

Command Window **Пример 3.5.3-74**

```
>>1/2+3/4
ans=    1.2500
>>sym(1/2+3/4)
ans=    5/4
>>pi/2
ans=    1.5708
sym(pi/2)
ans=    pi/2
>>sin(pi/3)
ans=    0.8660
>>sym(sin(pi/3))
ans=    sqrt(3/4)
>>exp(1)
ans=    2.7183
>>sym(exp(1))
ans=    6121026514868074*2^(-51)
>>
```

Необходимо обратить внимание на то, что **MatLab** производит вычисления с представлением чисел в формате с плавающей точкой двойной точности (**double**), в то время как пакет **Symbolic** стремится представить числа в наиболее точном виде, то есть в целом или рациональном виде.

Необходимо также обратить внимание и на то, что результат символьных преобразований отображается при выводе без отступа, которым сопровождается выдача иных результатов. Это позволяет сразу опознать его как символьный, в отличие от обычных численных результатов.

Для создания группы символьных объектов служит команда **syms:**

syms arg1 arg2 ...

Например, набор команд

Command Window **Пример 3.5.3-75**

```
>>a=sym('a ');
>>b=sym('b');
>>c=sym('c');
>>x=sym('x ');
>>
```

соответствует следующей команде

```
>>syms a b c x;
>>
```

Рассмотрим еще один пример на применение команды **syms.**

Command Window **Пример 3.5.3-76**

```
>>syms x y1 y2
>>y1=sin(x)^2;
>>y2=cos(x)^2;
>>y1+y2
ans=       sin(x)^2+cos(x)^2
>>
```

Для работы с действительными и комплексными числами существует ряд /возможностей. Прежде всего, это применение опций **'real'**, **'unreal'**, **'imag'**. Например,

Command Window **Пример 3.5.3-77**

```
>>x=sym('x' , real);
>>y=sym('y', real);
>>z=x+i*y
z=         x+i*y
>>real(z)
ans=       x
>>imag(z)
ans=       y
>>
```

Функции **real(z)** и **imag(z)** обеспечивают выделение действительной и мнимой частей комплексного числа **z**.

Снятие наложенных ограничений на возможные значения переменных, например, определенной выше действительной переменной **x**, осуществляется командой

Command Window **Пример 3.5.3-78**

```
>>sym 'x', 'unreal'
% или
>>syms x unreal
```

что делает переменную **x** не вещественной.

Из вышеприведенных примеров видно, что создание символьного выражения осуществляется командой

sym('символьное_выражение ')

Символьные выражения стоятся на основе переменных, констант, арифметических операторов и функций – как встроенных, так и определяемых пользователем. В выражениях могут использоваться арифметические операции. В общем случае это векторные и матричные операции. Например,

Command Window - **Пример 3.5.3-79**

```
>> sym x a b
>> f=(sin(x)+a)^2*(cos(x)+b^2/sqrt(abs(a+b))
f=(sin(x)+a)^2*(cos(x)+b^2/sqrt(abs(a+b))
```

В математических выражениях могут использоваться как обычные, так и символьные переменные. Функция **findsym(S)** позволяет выделить символьные переменные в составе выражения **S**. Она возвращает в алфавитном порядке список всех символьных переменных выражения **S**. При отсутствии таковых возвращается пустая строка. Например:

Command Window **Пример 3.5.3-80**

```
>>sym x, z, y
>> a =2; b = 4 ;
>> findsym (a*x^2 + b*y + z)
ans =      x, y, z
>>findsym(a + b + x + y + z,)
ans =      x, y, z
>>
```

Символьные переменные могут быть элементами матриц и векторов, причем с ними можно выполнять различные операции. Например,

Command Window **Пример 3.5.3-81**

```
>>sym a b c d
>>M=[a b; c d]
M=     [ a, b]
       [ c, d]
>> inv(M)
ans=   [ d/(a*d-b*c), -b/(a*d-b*c)]
       -c/(a*d-b*c),  a/(a*d-b*c)]
>>
```

MatLab в отличие от современных систем **MathCAD**, **Maple** или **Mathematica**, пока не способна выводить выражения и результаты их преобразований в естественной математической форме с использованием общепринятых спецзнаков для отображения интегралов, сумм, произведений и т. д. Тем не менее, некоторые ограниченные текстовым форматом возможности близкого к математическому виду вывода, обеспечивает функция **pretty(S)** . Она позволяет вывести выражения **S** в формате, приближенном к математическому выражению, например,

Command Window **Пример 3.5.3-82**

```
>> sym x a b
>> f=(sin(x)+a)^2*(cos(x)+b)^2/sqrt(abs(a+b))
f=(sin(x)+a)^2*(cos(x)+b)^2/sqrt(abs(a+b))
>>pretty(f)
                    2                     2
                (sin(x) + a)  (cos(x) + b)
                ------------------------------
                           1/2 | a + b |
>>
```

Символьные операции позволяют находить или точные значения выражения, или значения со сколь угодно большой точностью. Как известно, для преобразования значения числовой переменной в символьную служит функция **sym()**. Причем, при переходе от числовых к символьным выражениям по умолчанию используется запись чисел в виде рациональной дроби. Использование рациональных дробей при выполнении символьных вычислений означает, что всегда получается точный результат, не содержащий погрешность округления.

Для вычисления символьного выражения с заданной точностью предназначена функция **vpa( )**:

vpa(символьная _переменная, число значащих цифр)

По умолчанию результат этой функции содержит 32 значащие цифры.

Второй аргумент **vpa()** задает число значащих цифр только для данного вызова этой функции. Для глобальной установки применяется функция **digits(),** во втором аргументе которой указывается требуемое количество цифр.

Функция **digits()** служит для установки числа цифр в числах арифметики произвольной точности (по умолчанию 32).

Command Window **Пример 3.5.3-82**

```
>>digits
Digits = 32
>>vpa(pi)
ans =    3.1415926535897932384626433832795
>>digits(5)
>> pi
ans =    3.1416
>>vpa(pi,3)
ans =    3.14
>>
```

Для проведения вычислений в арифметике произвольной точности служит функция **vpa( ):**

- **R = vpa(S)** — возвращает результат вычислений каждого элемента символьного массива **S**, используя арифметику произвольной точности с текущим числом цифр **D**, установленным функцией **digits**. Результат **R** имеет тип **sym**.
- **vpa(S,D)** — возвращает результат вычислений каждого элемента массива **S**, используя арифметику произвольной точности с количеством знаков чисел **D**.

Command Window **Пример 3.5.3-83**

```
>>vpa(exp(1),50)
ans =    2.7182818284590450907955982984276488423347473144531
>>
```

3.5.3.3.3 Упрощение выражений, подстановки и разложение на множители

Для упрощения выражений в **MatLab** общего вида используются функции **simple()** и **simplify(S).** Функция **simplify(S)** реализует мощный алгоритм упрощения символьных выражении массива **S**, содержащих как тригонометрические, экспоненциальные и логарифмические функции, так и специальные. Алгоритм, заложенный в **simple(S),** пытается получить выражение, которое представляется меньшим числом символов, чем исходное. Если упрощение невозможно, то возвращается исходное выражение. Например:

Command Window **Пример 3.5.3-84**

```
>>syms a b x;
>>V=[sin(x)^2 + cos(x)^2   log(a*b)]
>>simplify(V)
ans =     [ 1, log(a*b)]
>>simplify((a^2 - 2*a*b + b^2) / (a - b))
ans =     a - b
>>
```

Для расширения (раскрытия скобок) выражений используется функция **expand(S).** Эта функция расширяет выражения, входящие в массив **S**. Рациональные выражения она раскладывает на простые дроби, полиномы — на полиномиальные разложения и т. д. Функция работает со многими алгебраическими и тригонометрическими функциями. Например:

Command Window **Пример 3.5.3-85**

```
>>syms a b x;
>>S=[(x + 2)*(x + 3)*(x + 4)    sin(2*x)];
>>expand(S)
ans =   [ x^3 + 9*x^2 + 26*x + 24 ,  2*sin(x)*cos(x)]
>>expand(sin(a + b))
ans =   sin(a)*cos(b) + cos(a)*sin(b)
>>expand((a + b)^3)
ans =   a^3 + 3*a^2*b + 3*a*b^2 + b^3
>>
```

Одной из самых эффектных и часто используемых операций символьной математики является операция подстановки. Она реализуется функцией **subs()**, имеющей несколько различных форм записи:

- **subs(S)** - заменяет в символьном выражении **S** все переменные их символьными значениями, которые берутся из вычисляемой функции или рабочей области системы **MatLab**.
- **subs(S,NEW)** - заменяет все свободные символьные переменные в **S** из списка **NEW**.
- **subs (S, OLD, NEW)** — заменяет **OLD** на **NEW** в символьном выражении **S**. При одинаковых размерах массивов **OLD** и **NEW** замена идет поэлементно. Если в **S OLD** - скаляры, а **NEW** - числовой массив или массив ячеек, то скаляры расширяются до массива результатов.

Пример 3.5.3-86

```
>> syms a  b  x  y;
>>subs(x - y, y, 1)
ans =    x - 1
>> subs(sin(x) + cos(y), [x,y], [a,b])
ans =    sin(a)+cos(b)
>>
```

1. **Обращение функции – finverse( )**
   Часто возникает необходимость в задании функции, обратной по отношению к заданной функции **f** . Для этого в **Symbolic** имеется функция обращения **finverse**, которая задается в двух формах:
   - **g = finverse(f)** – возвращает функцию, обратную к **f.** Считается, что **f** – функция одной переменной, например 'x'. Тогда **g(f(x)) = x**.
   - **g = finverse(f,v)** – возвращает функцию, обратную к f, относительно заданной переменной **v**, так что **g(f(v)) = v**. Эта форма используется, если **f** – функция нескольких переменных.

Пример 3.5.3-87

```
>> syms x
>> finverse(sinh(x))
ans =    asnnh(x)
>> finverse(exp(x))
ans =    log(x)
>>
```

2. **Суперпозиция функций – compose( )**
   compose(f, g) – возвращает **f(g(y)),** где **f = f(x)** и **g = g(y).** Независимые переменные **x** и **y** находятся с помощью функции **findsym( ).**
3. **Функция разложения в ряд Тейлора – taylor( )**
   - **taylor (f)** – возвращает шесть членов ряда Маклорена функции **f** ;
   - **taylor (f, n, x, a)** – возвращает *n* членов ряда Тейлора в точке **x = a** ;
   - **taylor (f, n)** – возвращает **(n - 1)** членов ряда Маклорена функции **f** ;
   - **taylor (f, a)** – возвращает шесть членов ряда Тейлора функции **f** в точке **a**.

Command Window **Пример 3.5.3-88**

```
>>x = sym('x')
>>taylor(sin(x))
ans =        x - 1/6*x^3 + 1/120*x^5
>>taylor(int(sin(x))
ans =       - 1 +1/2*x^2 - 1/24*x^4
>>
```

Для разложения выражения на простые множители используется функция **factor(S).** Эта функция поэлементно разлагает выражения вектора **S** на простые множители, а целые числа - на произведение простых чисел. Следующие примеры иллюстрируют применение функции:

Command Window **Пример 3.5.3-89**

```
>> x=sym('x');
>>factor(x^7-1)
ans =        ( x - 1 )*( x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 )
>>factor(x^2 - x - 1)
ans =        x^2 - x - 1
>>factor(sym('123456789'))
ans =        (3 )^2*(3803)*(3607)
>>
```

1. **Комплектование по степеням** – **collect( )**
   Функция **collect(S,v)** обеспечивает комплектование выражений в составе вектора или матрицы **S** по степеням переменной **v**.

2. **Упрощение выражений** – **simple( )**
   Функция **simple(S)** выполняет различные упрощения для элементов массива **S** и выводит как промежуточные результаты, так и самый короткий конечный результат. В другой форме – **[R, HOW] = simple(S)** – промежуточные результаты не выводятся.

3. **Приведение к рациональной форме** – **numden( )**
   Функция **[N,D] = numden(A)** преобразует каждый элемент массива **A** в рациональную форму в виде отношения двух неприводимых полиномов с целочисленными коэффициентами. При этом **N** и **D** — числители и знаменатели каждого преобразованного элемента массива.

Command Window **Пример 3.5.3-90**

```
>> [n,d] = numden(sym(8/10))
n =             4
d =             5
>>syms x  y
>> [ n, d ] = numden (x*y + y / x)
n =             y*(x^2 + 1)
d =             x
>>
```

4. **Функция вычисления суммы рядов** – **symsum( )**
   Для аналитического вычисления суммы ряда служит команда **symsum( ):**

- **symsum(S)** – возвращает символьное значение суммы бесконечного ряда по переменной, найденной автоматически с помощью функции **findsym( );**
- **symsum(S, v)** – возвращает сумму бесконечного ряда по переменной **v**;

- **symsum(S, a, b)** и **symsum(S, v, a, b)** – возвращают конечную сумму ряда в пределах номеров слагаемых от **a** до **b**.

Command Window **Пример 3.5.3-91**

```
>> [n,d] = numden(sym(8/10))
n =       4
d =       5
>>syms x  y
>> [ n, d ] = numden (x*y + y / x)
n =       y*(x^2 + 1)
d =       x
>>
>>x = sym('x');
>>symsum(x^2)
ans =            1/3*x^3-1/2*x^2+1/6*x
>>symsum([x, x^2, x^3], 1, 5)
ans =            [ 15,  55,  225]
>>
```

3.5.3.3.4 Разложение функций в ряды. Вычисление сумм и произведений членов ряда

Функция разложения в ряд Тейлора – **taylor( ).**

- **taylor (f)** – возвращает шесть членов ряда Маклорена функции **f** ;
- **taylor (f, n, x, a)** – возвращает **n** членов ряда Тейлора в точке x = a ;
- **taylor (f, n)** – возвращает **(n - 1)** членов ряда Маклорена функции **f** ;
- **taylor (f, a)** – возвращает шесть членов ряда Тейлора функции f в точке **a**.

Command Window **Пример 3.5.3-92**

```
>>x = sym('x')
>>taylor(sin(x))
ans =        x - 1/6*x^3 + 1/120*x^5
>>taylor(int(sin(x))
ans =        - 1 +1/2*x^2 - 1/24*x^4
>>
```

Для аналитического вычисления суммы членов ряда служит команда **symsum( )**:

- **symsum(S)** - возвращает символьное значение суммы бесконечного ряда по переменной, найденной автоматически с помощью функции **findsym( )**;
- **symsum(S, v)** - возвращает сумму бесконечного ряда по переменной **v**;
- **symsum(S, a, b)** и **symsum(S, v, a, b)** - возвращают конечную сумму ряда в пределах номеров слагаемых от **a** до **b**.

Command Window **Пример 3.5.3-93**

```
>> x = sym('x');
>> symsum(x^2)
ans =     1/3*x^3-1/2*x^2+1/6*x
>>symsum([x, x^2, x^3], 1, 5)
ans =     [ 15,  55,  225]
>>
```

Для аналитического вычисления произведений членов ряда служит команда:

Command Window **Пример 3.5.3-94**

```
>> x = sym('x');
>> symsum(x^2)
ans =     1/3*x^3-1/2*x^2+1/6*x
>>symsum([x, x^2, x^3], 1, 5)
ans =     [ 15,  55,  225]
>>
```

3.5.3.3.5 Вычисление пределов, производной и дифференциала

Напомним, что число **L** называется пределом функции **f(x)** в точке **a**, если при **x**, стремящимся к **a** (или **x** → **a**), значение функции неограниченно приближается к **L**. Это обозначается следующим образом:

lim f(x) = L.

Предел может быть конечным числом, положительной или отрицательной бесконечностью.

Имеется функции (например, разрывные в точке **x=a**), у которых нет предела в самой точке **x=a**, но есть предел при **x → a – 0** или при **x→ a + 0**, где под **0** подразумевается очень малое число. В первом случае говорят о существовании предела слева от точки **x=a**, а во втором – справа от этой точки. Если эти пределы равны, то существует предел функции в точке **x=a**.

Для вычисления пределов аналитически заданной функции **f(x)** служит функция **limit(),** которая может записываться в нескольких вариантах:

- **limit(f, x, a)** – возвращает предел символьного выражения **f** в точке **x → a**;
- **limit(f, a)** – возвращает предел для независимой переменной, определяемой функцией **findsym()**;
- **limit(f)** - возвращает предел при **a=0**;
- **limit(f, x, a, `right`)** или **limit(f, x, a, `left`)** – возвращает предел в точке **a** справа или слева.

Рассмотрим примеры применения этих функций.

Command Window **Пример 3.5.3-95**

```
>> syms a x
>> limit(sin(a*x)/(a*x))
ans =          1
>> limit(sin(a*x)/x)
ans =          a
>> limit(2*sin(x)/x)
ans =          2
>> limit(2+sin(x)/x,0)
ans =          3
>> limit(tan(x),pi)
ans =          0
>> limit(tan(x),pi/2)
ans =          NaN
>> limit(tan(x),x,pi/2,'right')
ans =           -Inf
>> limit(tan(x),x,pi/2,'left')
ans =           Inf
>>
```

Для вычисления в символьном виде производных от выражения **S** служит функция **diff()**, которая записывается в формате

diff(S,'v') или **diff(S, sym('v'))**.

Она возвращает символьное значение первой производной от символьного выражения или массива символьных выражений **S** по переменной **v**. Эта функция возвращает

$$S'(v) = \frac{dS}{dv}.$$

Для вычисления **n** –й производной используется следующие форматы:

- **diff(S, n)** — возвращает **n**-ю (**n** — целое число) производную символьного выражения или массива символьных выражений **S** по переменной **v**.
- **diff(S, v, n)** и **diff(S, n, v)** – возвращает **n**-ю производную **S** по переменной **v**.

Command Window **Пример 3.5.3-96**

```
>>x =sym( 'x' ); y=sym( 'y' );
>> diff(x^y)
ans =      x^y*y / x
>>slmplify(ans)
ans =      x^(y-1)*y
>>diff(s1n(y*x), x, 3)
ans =      - cos(y*x)*y^3
>>diff([x^3  sin(x)  exp(x)], x)
ans =      [ 3*x^2,   cos(x),  exp(x)]
>>
syms x
f = sin(5*x)
ans =        5*cos(5*x)

g = exp(x)*cos(x)
diff(g)
ans =         exp(x)*cos(x)-exp(x)*sin(x)
To take the second derivative of g, enter
diff(g,2)
ans =         -2*exp(x)*sin(x)
You can get the same result by taking the derivative twice:
diff(diff(g))
ans =         -2*exp(x)*sin(x)
Note that to take the derivative of a constant, you must first define the
constant as a symbolic expression. For example, entering
c = sym('5');
diff(c)
ans =         0
diff(5)
ans =         []
because 5 is not a symbolic expression.
```

3.5.3.3.6 Вычисление неопределенных и определенных интегралов

Функция **int( )** вычисляет неопределенные и определенные интегралы

- **int(S)** — возвращает символьное значение неопределенного интеграла от символьного выражения или массива символьных выражений **S** по переменной, которая автоматически определяется функцией **findsym**. Если **S** — скаляр или матрица, то вычисляется интеграл по переменной **'x'**.

- **int(S, v)** — возвращает неопределенный интеграл от **S** по переменной **v**.
- **int(S, a, b)** — возвращает определенный интеграл от **S** с пределами интегрирования от **a** до **b,** причем пределы интегрирования могут быть как символьными, так и числовыми.
- **int(S, v, a, b)** — возвращает определенный интеграл от **S** по переменной **v** с пределами от **a** до **b.**

Command Window Пример 7.3.8-1

```
>> int(sin(x)^3, x)
ans =     - 1/3*sin(x)^2*cos(x)-2/3*cos(x)
>> int(log(2*x), x)
ans =     log(2*x)*x - x
>> int((x^2-2)/(x*3-1), x, 1, 2)
ans =     -inf
>> int((x^2-2)/(x*3-1), x, 2, 5)
ans =    - 2/3*log(2)+ 2/3*log(31) + 2/3*3^(1/2)*atan(11/3*3^(1/2)) -...
         2/3*log(7) - 2/3*3^(1/2)*atan(5/3*3^(1/2))
>> int([x^3   sin(x)   exp(x)], x)
ans =     [ 1/4*x^4,  -cos(x),    exp(x)]
>> int(log(sin(x)),x,0,pi/2)
ans =      -pi/2*log(2)
>>
```

С помощью функции **int( )** можно вычислять имеющие аналитическое решение сложные интегралы, например с бесконечными пределами (или одним из пределов), а также кратные интегралы.

Command Window Пример 7.3.8-2

```
» int(log(1+exp(-x),x,0,inf)
ans =     pi^2/12
» syms x a b
» int(int(int(x^2 + y^2)*z, x, 0, a), y, 0, a), z, 0, a)
ans =     1/3*a^6
>>
```

3.5.3.3.7 Средства визуализации результатов символьных вычислений

Визулизация функции одной переменой осуществляется при помощи функции ezplot().

1. Графопостроитель – funtool

Команда **funtool** создает интерактивный графический калькулятор, позволяющий быстро построить две функции одной переменной - **f(x)** и **g(x).** Например, одна может задавать собственно функцию, а другая — ее производную. Функции обозначаются как **' f = '** и **' g = '** и после знака равенства можно набрать функции с помощью клавиш калькулятора в его нижней части. С помощью полей **'x = '** и **' a = '** можно задать диапазон изменения переменной **x** и значение масштабирующего параметра **a**.

При запуске команды **funtool** появляются окна для двух функций и окно калькулятора (рис. 5.3.1). По умолчанию заданы функции **f(x) = x** и **g(x) = 1**, предел изменения ***x*** от **-2π** до **2π** и ***a*** **= 1/2**.

Верхний ряд кнопок вычислителя относится только к функции *f(x)* и задает следующие операторы:

- **df/dx** — символьное дифференцирование **f(x)**;
- **int f** — символьное интегрирование **f(x)** при наличии замкнутой формы;

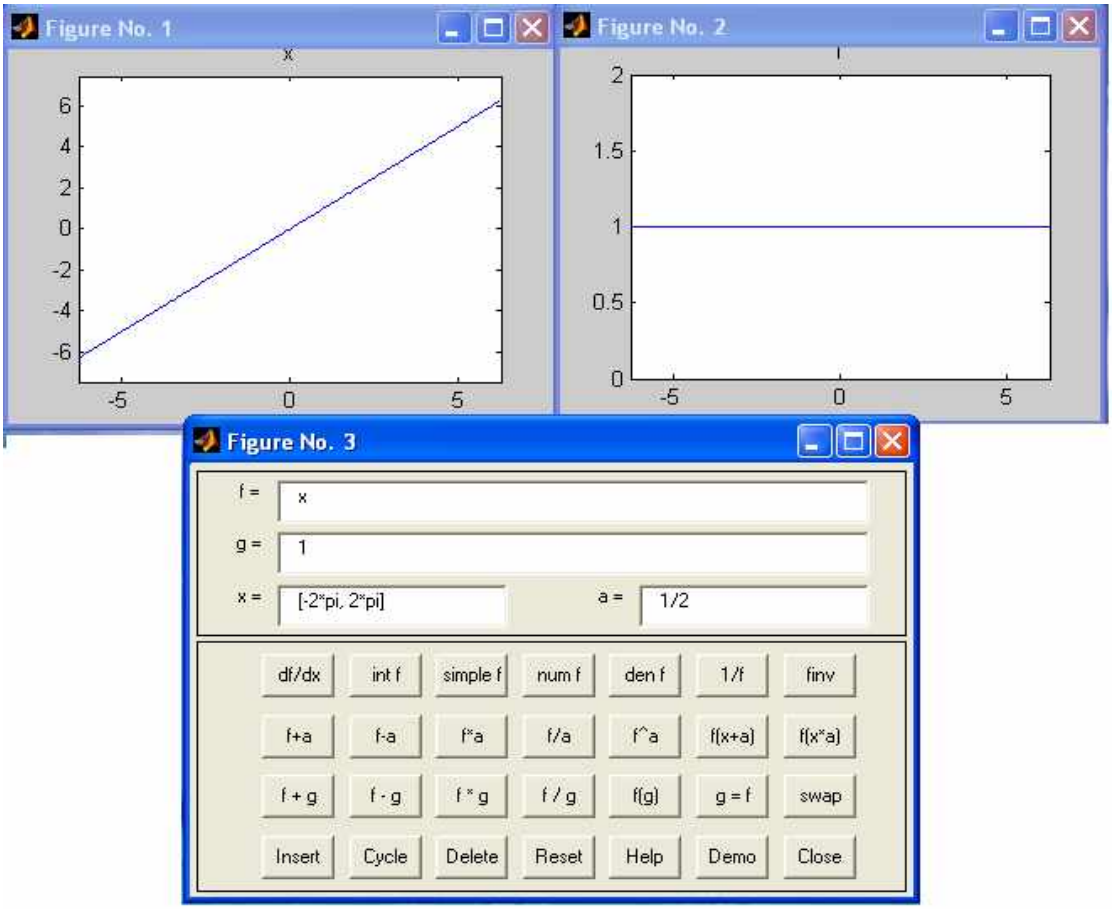


Рис. 7.3.9-1. Внешний вид графопостроителя funtool

- **simple f** – упрощение выражения, если таковое возможно;
- **num f** – выделение числителя рационального выражения;
- **den f** – выделение знаменателя рационального выражения;
- **1/f** – замена **f(x)** на **1 / f(x);**
- **finv** – замена **f(x)** инверсной функцией.
- второй ряд клавиш выполняет операции масштабирования и сдвига **f(x)** с применением параметра **'a'.**
- третий ряд клавиш предназначен для осуществления бинарных операций над функциями **f(x)** и **g(x).**
- четвертый ряд клавиш служит для работы с памятью калькулятора и иных операций:
- **Insert** – помещает текущую функцию в список функций.
- **Cycle** – выполняет текущую функцию из списка.
- **Delete** – удаляет выделенную функцию из списка.
- **Reset** – устанавливает **f, g, x, a** и **fxl i st** в исходное состояние.
- **Help** – выводит описание калькулятора.
- **Demo** – запускает демонстрационный пример.
- **Close** – завершает работу с калькулятором.

Благодаря описанным средствам, вычислитель позволяет задать инересующую вас функцию, выполнить ее преобразования (например, дифференцирование и интегрирование) и, наконец, построить график функции и результатов ее преобразования (рис.5.3.2)

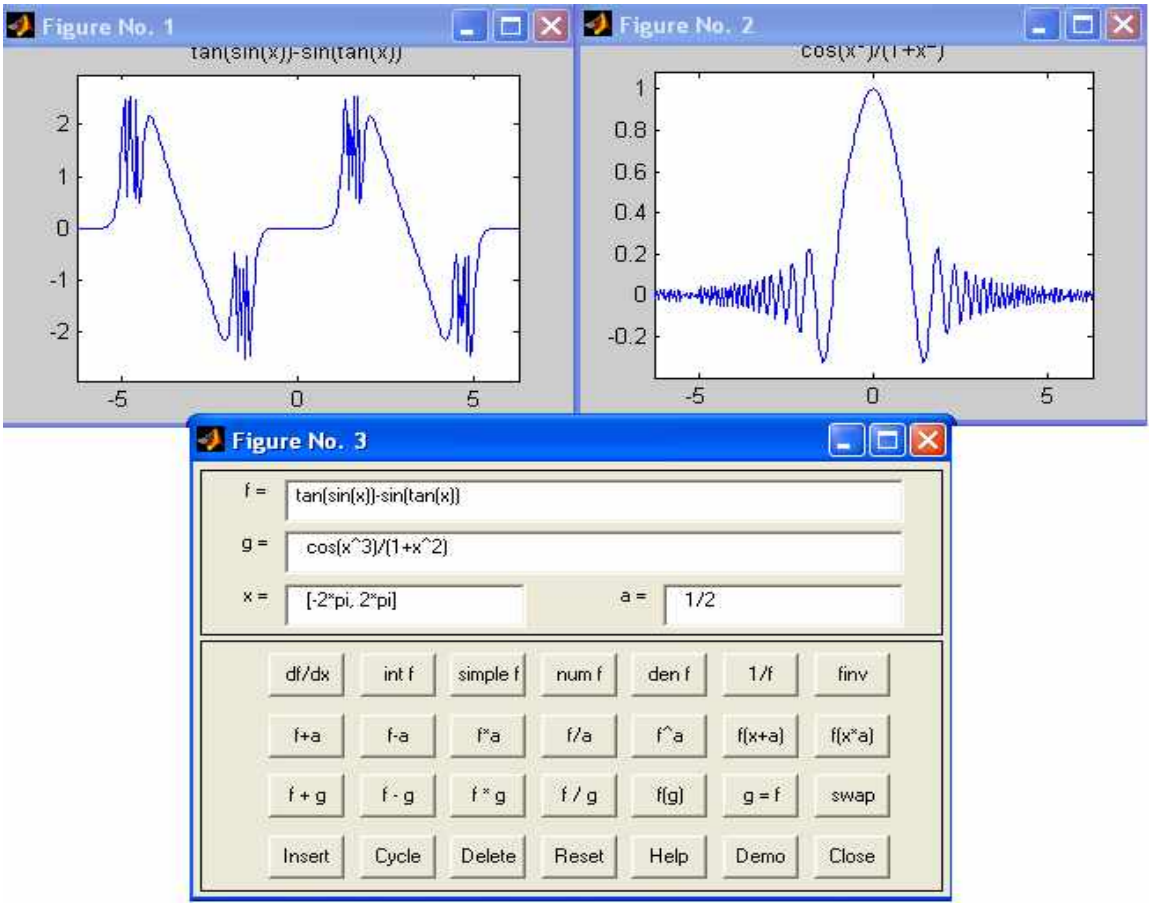


Рис. 7.3.9-2. Построения графиков некоторых функций

Таким образом, графопостроитель **funtool** является весьма удобным средством визуализации графиков самых различных функций.

2. Графики поверхностей – ezsurf( ) и ezsurfc( )

Команда **ezsurf** служит для построения графиков поверхностей, задаваемых функциями двух переменных **f(x, y):**

- **ezsurf(f)** – построение поверхности **f(x,y)** *с* параметрами **x** *и* **y**, меняющимися по умолчанию от **-2π** до **2π**;
- **ezsurf(f,domain)** – построение поверхности*f(x,y)* с пределами изменения ***x*** и ***y***, заданными параметром **domain**;
- **ezsurf(x,y,z)** – построение поверхности, заданной параметрически зависимостями **x(s, t), y(s, t), z(s, t)** при **s** и **t**, меняющихся в интервале от **-2π** до **2π**;
- **ezsurf(x,y,z,[smin, smax, tmin, tmax])** – построение поверхности, заданной параметрически зависимостями **x(s, t), y(s, t), z(s, t)** при **s** и **t** меняющихся в заданном интервале.

Следующий пример показывает действие этой команды:

Command Window – Пример 7.3.9-1

```
>>syms x y
>> ezsurf(real(asec(x+i*y)))
>>
```

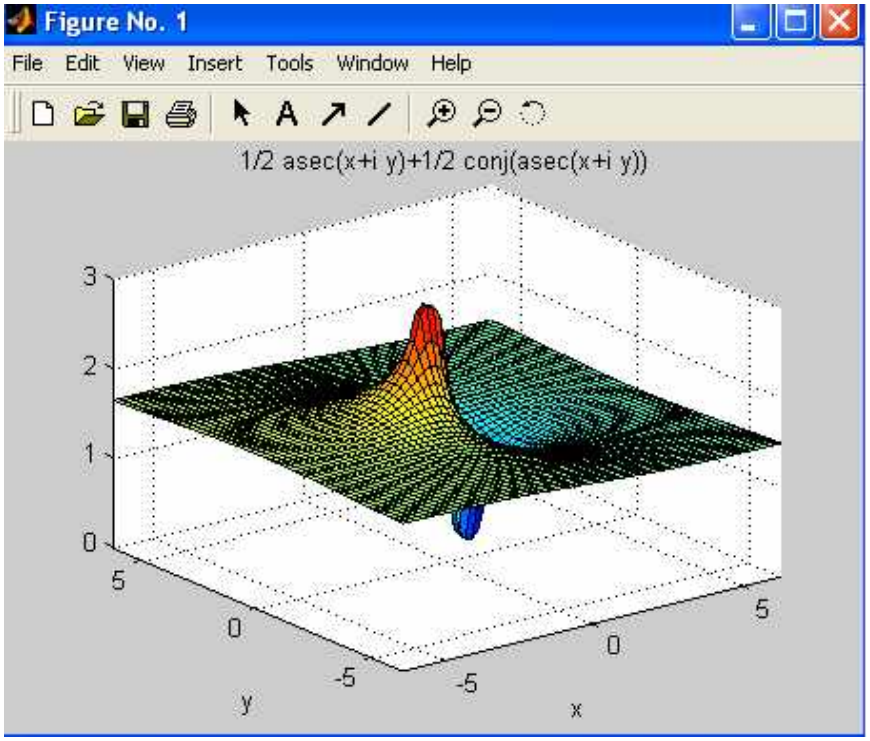


Рис. 7.3.9-3. Пример построения графика поверхности командой **ezsurf**

Аналогичная по синтаксису записи группа команд **ezsurfc( )** строит еще и контурный график поверхности на плоскости, лежащей под поверхностью.

3.5.3.4. M-файлы и программирование средствами MatLab

3.5.3.4.1. Основные понятия и средства программированиям в MatLab

До сих пор мы в основном использовали систему **MatLab** в режиме непосредственного счета — *в* **командном режиме**. Однако при решении серьезных задач возникает необходимость сохранения используемых последовательностей вычислений, а также их дальнейшей модификации. Иными словами, существует необходимость **программирования** решения задач.

Это может показаться отходом от важной цели, которая преследуется разработчиками большинства математических систем, — выполнения математических вычислений без использования традиционного программирования. Однако это не так. Выше было показано, что множество математических задач решается в системе MATLAB без программирования. С использованием языков высокого уровня для их решения потребовалось бы написать и оттестировать сотни программ.

Практически невозможно предусмотреть в одной, даже самой большой и мощной, математической системе возможность решения всех задач, которые могут интересовать пользователя. Программирование в системе **MatLab** является эффективным средством ее расширения и адаптации к решению специфических проблем. Оно реализуется с помощью **языка программирования системы MatLab**.

Большинство объектов этого языка, в частности все команды, операторы и функции, одновременно являются объектами **входного языка общения с системой в командном режиме работы**. Так что фактически мы приступили к описанию языка программирования системы **MatLab** с первых строк данной книги.

Так в чем же отличие входного языка от языка программирования? В основном — в способе фиксации создаваемых ими кодов. Сессии в командном режиме работы не сохраняются в памяти компьютера (ведение дневника не в счет). Хранятся только определения созданных в ходе их выполнения переменных и функций. А вот программы на языке программирования **MatLab** сохраняются в виде текстовых **m-файлов**. При этом могут сохраняться как целые программы в виде **файлов-программ**, так и отдельные **программные модули — функции**. Кроме того, важно, что программа может менять структуру алгоритмов вычислений в зависимости от входных данных и данных, создаваемых в ходе вычислений.

С позиций программиста язык программирования системы является типичным **проблемно-ориентированным языком программирования высокого уровня**. Точнее говоря, это даже язык сверхвысокого уровня, содержащий сложные операторы и функции, реализация которых на обычных языках (например, Бейсике, Паскале или Си) потребовала бы много усилий и времени. К таким функциям относятся матричные функции, функции быстрого преобразования Фурье и др., а к операторам — операторы построения разнообразных графиков, генерации матриц определенного вида.

Итак, программами в системе **MatLab** являются **m-файлы** текстового формата, содержащие запись программ в виде программных кодов. Язык программирования системы **MatLab** имеет следующие средства:

- данные различного типа;
- константы и переменные;
- операторы, включая операторы математических выражений;
- встроенные команды и функции;
- функции пользователя;
- управляющие структуры;
- системные операторы и функции;
- средства расширения языка.

Тексты программ в системе **MatLab** пишутся на языке высокого уровня, достаточно понятном для пользователей умеренной квалификации в области программирования. Язык программирования **MatLab** является типичным **интерпретатором**.

Интерпретация означает, что **MatLab** не создает исполняемых конечных программ. Они существуют лишь в виде **m-файлов**, для выполнения которых необходима среда **MatLab**. Однако для программ на языке **MatLab** созданы компиляторы, транслирующие программы **MatLab** в коды языков программирования **C** и **C++.** Это решает задачу создания исполняемых программ, первоначально разрабатываемых в среде **MatLab**. Компиляторы для системы **MatLab** являются вполне самостоятельными программными средствами.

Начальное представление о **переменных, встроенных константах и функциях** уже было дано в предшествующих главах.

В **MatLab** определены следующие основные типы данных, в общем случае представляющих собой многомерные массивы:

- **single** — числовые массивы с числами одинарной точности;
- **double** — числовые массивы с числами удвоенной точности;
- **char** — строчные массивы с элементами-символами;
- **sparse** — наследует свойства double, разреженные матрицы с элементами-числами удвоенной точности;
- **cell** — массивы ячеек; ячейки, в свою очередь, тоже могут быть массивами;
- **struct** — массивы структур с полями, которые также могут содержать массивы;
- **function_handle** — дескрипторы функций:
- **int32, uint32** — массивы 32-разрядных чисел со знаком и без знаков;
- **int16,uint16** — массивы 16-разрядных целых чисел со знаком и без знаков;

- **int8, uint8** — массивы 8-разрядных целых чисел со знаками и без знаков.

Каждому типу данных можно соотнести некоторые характерные для него операции, называемые **методами**. Поскольку в иерархии типов данных сверху находятся данные типа **array**, это значит, что все виды данных в **MatLab** являются массивами.

Язык программирования системы **MatLab** вобрал в себя почти все средства, необходимые для реализации различных технологий программирования:

- процедурного;
- операторного;
- функционального;
- логического;
- структурного (модульного);
- объектно-ориентированного;
- визуально-ориентированного.

В основе **процедурной**, **операторной** и **функциональной** технологии программирования лежат **процедуры**, **операторы** и **функции**, используемые как основные объекты языка. Эти типы объектов присутствуют в **MatLab**. **Логическое** программирование реализуется в **MatLab** с помощью **логических операторов** и **функций**.

Наиболее ярко в **MatLab** представлены идеи **структурного программирования**. Подавляющее большинство функций и команд языка представляют собой вполне законченные модули, обмен данными между которыми происходит через их входные параметры, хотя возможен обмен информацией и через глобальные переменные. Программные модули оформлены в виде текстовых **m-файлов**, которые хранятся на диске и подключаются к программам по мере необходимости. Важно отметить, что в отличие от многих языков программирования, применение тех или иных модулей не требует предварительного объявления, а для создания и отладки самостоятельных модулей **MatLab** имеет все необходимые средства. Подавляющее большинство команд и функций системы **MatLab** поставляется в виде таких модулей.

Объектно-ориентированное программирование также широко представлено в системе **MatLab**. Оно особенно актуально при программировании задач графики. Что качается **визуально-ориентированного** программирования, то в **MatLab** оно представлено в основном в пакете моделирования заданных блоками устройств и систем **Simulink**.

Здесь необходимо отметить, что для языка системы **MatLab** различие между **командами** (выполняемыми при вводе с клавиатуры) и **программными операторами** (выполняемыми из программы) является условным. И команды, и программные операторы могут выполняться как из программы, так и в режиме прямых вычислений.

В общем виде **функция** преобразует одни данные в другие. Для многих функций характерен **возврат значений** в ответ на обращение к ним с указанием списка **входных параметров** — **аргументов**. Например, говорят, что функция **sin(x)** в ответ на обращение к ней возвращает значение синуса аргумента **x**. Поэтому функцию можно использовать в арифметических выражениях, например **2*sin(x+1)**. Для операторов (и команд), не возвращающих значения, такое применение обычно абсурдно.

Важное фактором является **двойственность операторов** и **функций**. Многие операторы имеют свои аналоги в виде функций. Так, например, оператор «+» имеет аналог в виде функции **sum().** Команды, записанные в виде ***Command argument*** нередко имеют форму записи и в виде функции ***Command(' argument') .***

Указанная **двойственность** лежит в основе выбора между процедурным и функциональным подходами к программированию, каждый из которых имеет своих поклонников и противников и может (в той или иной мере) подходить для решения различных классов задач. При этом переход от одного подхода программирования к другому

возможен в пределах одной программы и происходит настолько естественно, что большинство пользователей даже не задумывается над тем, каким же подходом (или стилем) программирования они преимущественно пользуются.

Имеющиеся в языке **MatLab** управляющие структуры: условных операторы **if... else...elseif...end, case,** циклы **for...end** и **while...end**, похожи на те, которые используются в языках **Бейсик** и **Паскаль .**

Программирование простых задач в среде **MatLab** очень напоминает программирование на языке **Basic**. Во многих случаях программы на языке **Basic** можно почти дословно перевести на язык системы, учтя небольшие отличия в синтаксисе этих языков.

3.5.3.4.2. m-файлы, программ и функций

Итак, мы установили, что работа в командном режиме (сессия) не является программированием. Внешним атрибутом последнего в **MatLab** служит задание последовательности действий по программе, записанной в виде **m-файла**. Для создания **m-файлов** может использоваться как редактор, встроенный в **MatLab**, так и любой текстовый редактор, поддерживающий формат **ASCII**. Подготовленный и записанный на диск **m-файл** с соответствующем именем становится частью системы, и его можно вызывать как из командной строки, так и из другого **m**-файла. Имеется два типа **m**-файлов: **файлы-программы** и **файлы-функции**. Важно, что в процессе своего создания они проходят синтаксический контроль с помощью встроенного в систему **MatLab редактора/отладчика m-файлов**.

Файл-программа, именуемый также **Script-файлом**, является просто записью последовательности команд без входных и выходных параметров. Он имеет следующую структуру:

```
Editor-Имя_файла
% Основной комментарий
% Дополнительный комментарий
    Тело файла с любыми выражениями
```

Файлы-программы имеют следующие особенности:

- они не имеют входных и выходных аргументов;
- работают с данными из рабочей области;
- в процессе выполнения не компилируются;
- представляют собой зафиксированную в виде файла последовательность операций, полностью аналогичную той, что используется в сессии.

Основным комментарием является первая строка текстовых комментариев, а дополнительным - последующие строки комментариев. Это связано с тем, что основной комментарий выводится при выполнении команд **lookfor** и **help имя_каталога**. Полный комментарий выводится при выполнении команды **help Имя_файла**. В качестве примера рассмотрим файл-программу **PlotSin** и пример вызова ее.

Editor -Пример_5_3_2_1

```
%  Построение  графика синусоиды линией красного цвета
%  с масштабной сеткой в интервале [xmin.xmax]
    x=xmin:0.1:xmax;
   plot(x.sin(x).'r')
   grid on
```

Command Window Пример 5.3.2-1

```
>> xmin=-1;
>> xmax=1;
>> Пример_5_3_2_1;
>>
```

Первые две строки файла **Пример_5_3_2_1 –** это комментарий, остальные – тело файла. Знак **%** в комментариях должен начинаться с первой позиции строки. В противном случае команда **help** ***name_файла*** не будет воспринимать комментарий и возвратит сообщение вида

No help comments found in-name.m.

Обратите внимание на то, что такой файл нельзя запустить без предварительной подготовки, сводящейся к заданию значений переменным **xmin** и **xmax**, которые используются в теле файла. Это следствие первого свойства файлов-программ - они работают с данными из рабочей области. Переменные, используемые в файлах-программах, являются глобальными, т. е. они действуют одинаково в командах сессии и внутри программного блока, которым является файл-сценарий. Поэтому заданные в сессии значения переменных используются и в теле файла. Имена файлов-программ нельзя использовать в качестве параметров функций, поскольку файлы-программы не возвращают значений. Можно сказать, что файл-программа – это простейшая программа на языке программирования **MatLab** .

Результаты работы Примера 5.3.2-1. приведены на рис. 5.3.2-1.

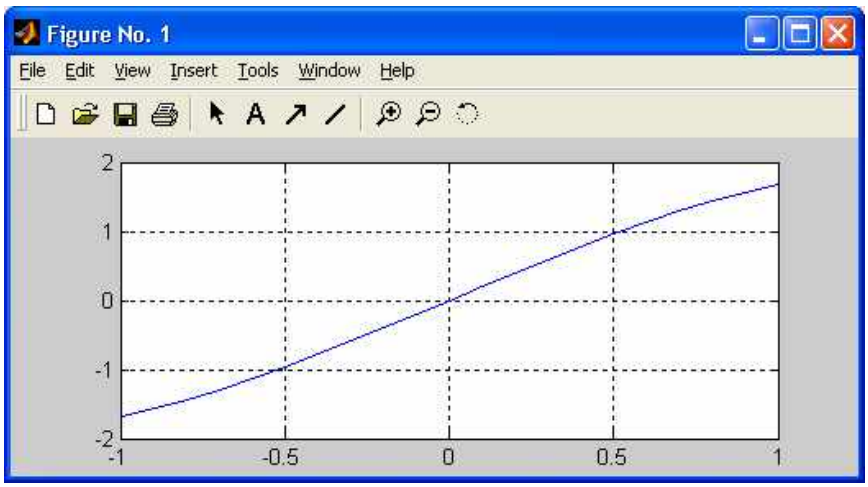


Рис.5.3.2-1. Результат работы файла-программы Пример_5_3_2_1

m-функция является типичным объектом языка программирования системы **MatLab**. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует

аппарат локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

Editor **имя-файла**

```
function var=f_name(Список_параметров)
% Основной комментарий
% Дополнительный комментарий
      Тело m-файла
      var=выражение
```

m-функции имеют следующие особенности:

- он начинается с объявления **function**, после которого указывается имя переменной **var** - выходного параметра, имя самой функции и список ее входных параметров;
- функция возвращает свое значение и может использоваться в виде **f_name (список_параметров)** в математических выражениях;
- все переменные, имеющиеся в теле файла-функции, являются **локальными**, т. е. действуют только в пределах тела функции;
- файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;
- правила вывода комментариев те же, что у файлов-программ;
- файл-функция служит средством расширения системы **MatLab**;
- при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы **MatLab**.

Последняя конструкция **var=выражение** вводится, если требуется, чтобы функция возвращала результат вычислений.

Приведенная форма файла-функции характерна для функции с одним выходным параметром. Если выходных параметров больше, то они указываются в квадратных скобках после слова **function**. При этом структура модуля имеет следующий вид:

Editor **имя-файла**

```
function [var1,var2....]=f_name(Список_параметров)
% Основной комментарий
% Дополнительный комментарий
      Тело файла с любыми выражениями
      var1=выражение
      var2=выражение
```

Такая функция во многом напоминает процедуру. Ее нельзя слепо использовать непосредственно в математических выражениях, поскольку она возвращает не единственный результат, а множество результатов — по числу выходных параметров.

Если функция используется как имеющая единственный выходной параметр, но имеет ряд выходных параметров, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в математических вычислениях. Поэтому, как отмечалось, данная функция используется как отдельный элемент программ вида:

[var1,var2,...]=f_name(Список_параметров)

После его применения переменные выхода **varl, var2,...** становятся определенными и их можно использовать в последующих математических выражениях и иных сегментах

программы. Если функция используется в виде **f_name(Список_параметров)**, то возвращается значение только первого выходного параметра — переменной **varl**.

Переменные, указанные в списке параметров функции, являются **локальными** и служат для переноса значений, которые подставляются на их место при вызовах функций. Эта особенность переменных-параметров хорошо видна при разборе примера 5.3.3, результат работы которого показан на рис. 5.3.3.

Editor **Пример_5_3_2_2**

```
function z=fun(x,y)
% Пример 5.3.2-2
     z=x^2+y^2;
```

Command Window **Пример 5.3.2-2**

```
>> z=fun(2,3);
>> x=
          0
>> y=
          0
>>z=
          13
>>
```

В этом примере в окне редактора создана функция **fun(x,y)** двух переменных **x** и **y**, вычисляющая $z = x^2 + y^2$. Поскольку переменные **x** и **y** указаны как параметры функции **fun(x, y),** то они являются локальными. В примере вне тела функции им заданы нулевые значения. Очевидно, что при вычислении значения **fun(2, 3)** в теле функции задается **x=2** и **y=3**. Поэтому результат - **z=13**. Однако после выхода из тела функции переменные **x** и **y** принимают свои исходные значения, равные нулю. Так что эти переменные меняют свои значения на значения параметров функции только локально — в пределах тела функции. А каков статус переменной **z** в нашем примере? Она, как и любая переменная, определенная в теле функции, также будет локальной. Изначально ее значение не определено. В теле функции переменная принимает значение **z=13**. А после возврата из функции, как нетрудно увидеть из примера 5.3.3, переменная **z,** несмотря на ее применение в теле функции, остается неопределенной. На это указывает сообщение, отображаемое после попытки вывода значения переменной **z**.

Возврат из функции производится после обработки всего тела функции, т. е. при достижении конца файла функции. При использовании в теле функции условных операторов, циклов или переключателей иногда возникает необходимость осуществить возврат функции раньше, чем будет достигнут конец файла. Для этого служит команда **return**. В любом случае, результатом, возвращаемым функцией, являются значения выходных параметров (в нашем случае выходным параметром является переменная **z**), присвоенные им на момент возврата.

У нашей функции имеется один недостаток — вывод на индикацию значения **z=13** из тела функции, хотя после этого **z** остается равным **0**. Чтобы убрать побочный эффект вывода значения **z**, достаточно установить знак **;** после математического выражения, определяющего **z**.

Этот пример наглядно показывает, что пропуск любого слова или даже простого оператора (вроде знака ;) может привести к не сразу понятным побочным эффектам и даже неверной работе функции. Программирование требует особой точности и педантичности.

Из сказанного ясно, что переменные в файлах-программах являются **глобальными**, а в файлах-функциях - **локальными**. Нередко применение глобальных переменных в

программных модулях может приводить к побочным эффектам. Применение локальных переменных устраняет эту возможность и отвечает требованиям структурного программирования.

Однако передача данных из модуля в модуль в этом случае происходит только через входные и выходные параметры, что требует тщательного планирования такой передачи. Однако и при создании файлов-функций порой желательно применение глобальных переменных. Ответственность за это должен брать на себя программист, создающий программные модули.

Команда

global varl var2...

позволяет объявить переменные модуля-функции глобальными. Таким образом, внутри функции могут использоваться и такие переменные, если это нужно по условиям решения вашей задачи. Причем, чтобы несколько программных модулей могли совместно использовать глобальную переменную, она должна быть объявлена как **global** во всех модулях.

Часто в ходе вычислений возникают ошибки. Например, при вычислении функции **sin(x)/x** - при $x = 0$ имеет место ошибка вида «деление на ноль». При появлении ошибки вычисления могут завершиться досрочно с выводом сообщения об ошибке. Следует, однако, отметить, что не все ошибки вызывают остановку вычислений. Некоторые сопровождаются только выдачей предупреждающей надписи. Такие ситуации должны учитываться программистом, отмечаться как ошибочные и по возможности устраняться. Для вывода сообщения об ошибке служит команда

error('Сообщение об ошибке');

при выполнении, которой вычисления прерываются, и выдается сообщение об ошибке, заданное в апострофах. Ниже дан пример вычисления функции **sd(x)=sin(x)/x**, в котором задано сообщение об ошибке:

Editor Пример_5_3_2_3

```
function f=sd(x)
if x==0 error('Ошибка - деление на 0'). end
f=sin(x)/x ;
```

Для выявления ситуации об ошибке использован оператор условного перехода if, который будет описан детально несколько позднее. Результат выполнения данной функции приводится ниже:

Command Window Пример 5.3.2-3

```
>>x=0;
>>f=sd(x);
???Ошибка - деление на 0
>>x=1;
>>f=sd(x);
>> f=

>>
```

Если остановка программы при появлении ошибки нежелательна, то может использоваться команда вывода предупреждающего сообщения

warning('Предупреждающее сообщение').

Эта команда выводит стоящее в апострофах сообщение, но не препятствует дальнейшей работе программы. Признаком того, что является ошибкой, а что - предупреждением, являются символы **???** и слово **Warning** в соответствующих сообщениях.

При создании функций со специальными свойствами весьма полезны две приведенные ниже функции:

- **nargin** — возвращает число входных параметров данной функции;
- **nargout** — возвращает число выходных параметров данной функции.

Пусть, к примеру, мы хотим создать функцию, вычисляющую сумму квадратов пяти аргументов **xl, x2, x3, x4 и x5**.
Обычный путь состоит в следующем — создаем функцию с именем **sum2_5**:

Editor `sum2_5_Пример_5_3_2_4`

```
function f=sum2_5(x1,x2,x3,x4,x5);
f=x1^2+x2^2+x3^2+x4*2+x5^*2;
```

Command Window `Пример 5.3.2-4`

```
» sum2_5_Пример_5_3_2_4 (1,2,3,4,5)
ans =
        55
» sum2_5(1,2)
??? Input argument 'x3' is undefined.
Error in ==> C:\MATI_AB\bin\sum2_5.m
On line 2 ==> f=x1^2+x2^2+x3^2+x4^2+x5^2;
>>
```

Итак, при наличии всех пяти аргументов функция работает корректно. Но если аргументов менее пяти, она выдает сообщение об ошибке. С помощью функции **nargin** можно создать функцию **sum2_5m**, которая работает корректно при любом числе заданных входных аргументов в пределах от **1** до **5**:

Editor `sum2_5_Пример_5_3_2_5`

```
f unction f=sum2_5m(x1 ,x2 , x3 ,x4 , x5) ;
n=nargin;
if n==1 f=x1^2; end
if n==2 f=x1^2+x2^2;end
if n==3 f=x1^2+x2^2+x3^2; end
if n==4 f=x1^2+x2 ^ 2+x3^2+x4 ^ 2: end
if n==5 f=x1^2+x2^2+x3^2+x 4^2+x5^2
```

В данной функции используется условный оператор **if...end**, который будет детально описан далее. Но и без этого ясно, что благодаря применению функции **nargin** и условного оператора, вычисления всякий раз идут по формуле с числом слагаемых, равным числу входных аргументов - от одного до пяти. Это видно из приведенных ниже примеров:

Command Window `Пример 5.3.2-5`

```
» sum2_5m(1)
ans =
        1
» sum2_5m(1,2)
ans =
        5
» sum2_5m( 1,2,3)
ans =
        14
» sum2_5m(1,2,3,4)
ans =
        30
» sum2_5m(1,2,3,4,5)
ans=
       55
» sum2_5m(1,2,3,4,5,6)
??? Error using ==> sum2_5m
Too many input arguments.
>>
```

Итак, при изменении числа входных параметров от **1** до **5** вычисления проходят корректно. При большем числе параметров выводится сообщение об ошибке. Это уже действует встроенная в интерпретатор **MatLab** система диагностики ошибок

Как отмечалось, команда **help name**, где **name** — имя **m-файла,** обеспечивает чтение первой строки с текстовым комментарием и тех строк с комментариями, которые следуют непосредственно за первой строкой. Комментарий, расположенный за пределами этой области, не выводится. Это позволяет создавать невыводимый программный комментарий.

Пустая строка прерывает вывод комментария при исполнении команды **help name**. Команда **type name** выводит текст программы со всеми комментариями, в том числе и следующими после пустых строк.

Команда **help catalog**, где **catalog** — имя каталога с **m-файлами**, позволяет вывести комментарий, общий для всего каталога. Такой комментарий содержится в файле **contents.m**, который пользователь может создать самостоятельно с помощью редактора **m-файлов**. Если такого файла нет, то будет выведен список первых строк комментариев для всех m-файлов каталога.

m-файлы-функции могут использоваться как в командном режиме, так и вызываться из других **m-файлов**. При этом необходимо указывать все входные и выходные параметры. Исключением является случай, когда выходной параметр единственный — в этом варианте функция возвращает единственный результат и может использоваться в математических выражениях. При использовании глобальных переменных они должны быть объявлены во всех **m-файлах**, используемых в решении заданной задачи, и во всех входящих в них встроенных подфункциях.

Имена функций должны быть уникальными. Это связано с тем, что при обнаружении каждого нового имени **MatLab** проверяет, относится ли это имя к переменной, подфункции в данном **m-файл**е, частной функции в каталогах **PRIVATE** или функции в одном из каталогов пути доступа. Если последняя встречается, то будет исполнена именно эта функция.

Если аргумент функции используется только для вычислений и его значения не меняются, то аргумент передается ссылкой, что уменьшает затраты памяти. В других случаях аргумент передается значением. Для каждой функции выделяется своя (рабочая) область памяти, не входящая в область, предоставляемую системе **MatLab**. Глобальные

переменные принадлежат ряду областей памяти. При их изменении меняется содержимое всех этих областей.

При решении задач с большим объемом данных может ощущаться нехватка оперативной памяти. Признаком этого становится появление сообщения об ошибке **«Out of memory».**

В этом случае может быть полезным применение следующих мер:

- стирание ставших ненужными данных, прежде всего больших массивов;
- увеличение размеров файла подкачки **Windows**;
- уменьшение размера используемых данных;
- снятие ограничений на размеры используемой памяти;
- увеличение объема физической памяти компьютера.

Чем больше емкость **ОП** компьютера, на котором используется система **MatLab** тем меньше вероятность возникновения указанной ошибки. Опыт показывает, что даже при решении задач умеренной сложности емкость **ОЗУ** не должна быть менее **256 Мбайт**.

3.5.3.4.3. Работа с m-файлами

Во время работы в **MatLab** часто необходимо создавать или редактировать **m-файлы**, а после этого возвращаться в командное окно **MatLab** для отладки или вычислений. Для этого имеется специальный редактор/отладчик, в котором можно исправлять текст и выполнять пошаговую отладку программы. После исправления необходимо сохранить сделанные изменения. Причем **m-файлы**, с которыми работают, должны быть доступны. Для этого либо текущая директория должна быть директорией с вашими файлами, либо необходимо проложить туда путь. Это можно сделать с помощью пункта меню **File/Set Path**, который позволяет сделать директории доступными.

Для поиска **m-файлов** система **MatLab** использует механизм путей доступа, поскольку **m-файлы** записываются в каталоги или папки файловой системы. Например, при поиске файла с именем **foo, MatLab** выполняет следующие действия:

- просматривает, не является ли **foo** именем переменной;
- просматривает, не является ли **foo** встроенной функцией;
- ищет в текущем каталоге **m-файл** с именем **foo.m**;
- ищет **m-файл** с именем **foo.m** во всех каталогах **списка путей доступа.**

Реально применяемые правила поиска являются более сложными. Однако приведенный выше упрощенный порядок поиска точно отражает механизм поиска **m-файлов**, с которыми обычно работает пользователь.

В процессе сеанса работы можно вывести на дисплей или внести изменения в список путей доступа, используя следующие функции:

- **path** - выводит на экран список путей доступа;
- **path(s)** - заменяет существующий список списком **s**;
- **addpath/home/lib** и **path(path, '/home/lib')** -добавляют новый каталог текущего подкаталога в список путей доступа;
- **rmpath /home/lib** -удаляет путь **/home/lib** из списка.

Список путей доступа, используемый по умолчанию, определен в файле **pathdef.m**, который размещен в каталоге **local**; этот файл выполняется при каждом запуске системы **MatLab**.

Кроме работы из командной строки, существует средство просмотра путей доступа **Path Browser**, которое поддерживает удобный графический интерфейс для просмотра и изменения списка путей.

Система **MatLab** использует понятие **текущего каталога** при работе с **m**-файлами во время сеанса работы. **Начальный текущий каталог** определен в файле запуска.

Для вывода текущего каталога на экран предназначена команда **cd**. Для изменения текущего каталога следует использовать команду

cd <новый путь доступа>.

Как было указано выше, при работе в системе **Windows** имеется специальное средство для просмотра и изменения путей доступа **Set Path** (рис. 5.3.3-1). Показанное далее окно открывается либо из меню **File/Set Path** основного окна, либо с помощью кнопки на инструментальной панели.

После дополнения списка путей доступа необходимо сохранить новый путь с помощью пункта меню **File/Save Path**, в противном случае установленный путь будет известен системе только на время одного сеанса работы.

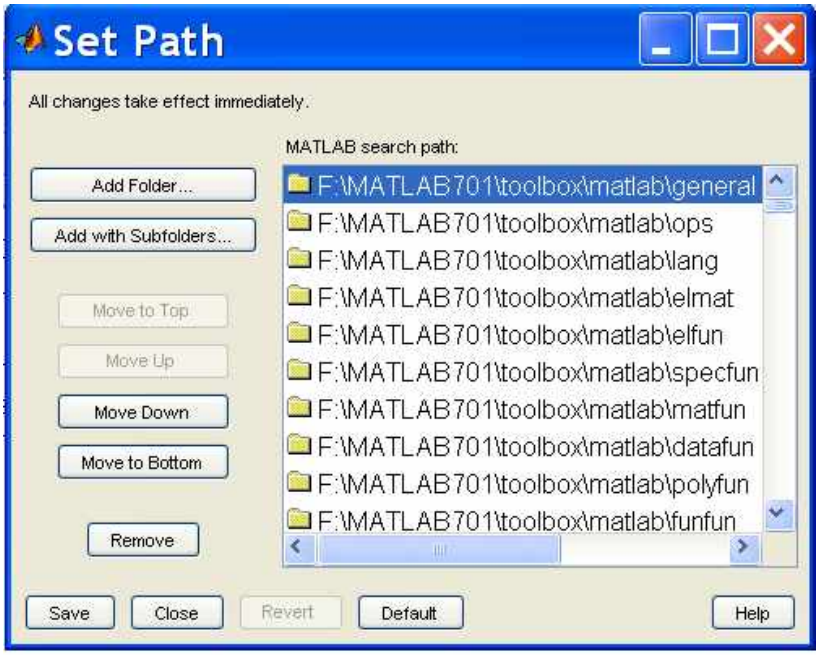


Рис. 5.3.3-1. Окно **Set Path**

Редактор/отладчик предоставляет из себя как средства редактирования текста **m-файла**, так и средства пошаговой его отладки. Один из способов вызова редактора – вызов из командной строки **MatLab** с помощью команды **edit**. Например, команда **edit poof** откроет встроенный редактор для редактирования файла **poof.m**, если в меню **File** в диалоговом окне **Preferences** не установлен вами другой редактор.

Можно открыть редактор и другим способом – с помощью меню **File/New** или кнопки **New File** на панели инструментов. Для открытия существующего **m-файла** выберите пункт **File/Open** или щелкните на кнопке **Open File**.

После вызова редактор/отладчик будет иметь вид, показанный на рис. 5.3.3-2.

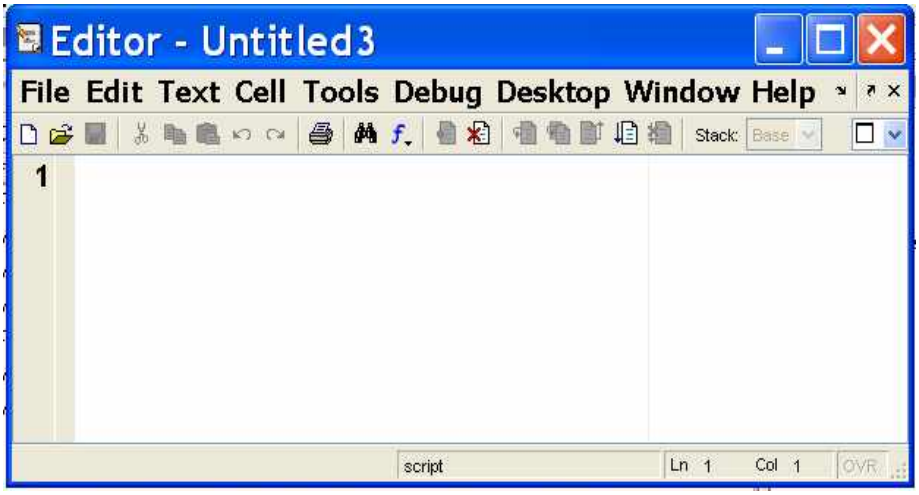


Рис. 5.3.3-2. Общий вид редактора/отладчика

Редактор, используемый в системе, имеет синтаксическую раскраску, т.е. слово или символ по мере ввода приобретают тот цвет, который соответствует их типу. Редактор различает такие типы вводимых слов:

- комментарии
- ключевые слова
- незаконченные строки
- законченные строки
- другой текст.

С помощью пункта меню **Tools/Fonts** можно настроить такие важные параметры, как используемый шрифт. В основном данный редактор не отличается от обычного многооконного текстового редактора - в нем работают все редактирующие клавиши (**Del, Bspace, Home** и т.д.). При редактировании файлов вы можете непосредственно перейти к требуемой строке при помощи пункта меню **Edit/GoTo Line** и указать номер требуемой строки в появившемся окне. После редактирования файла и повторного его запуска из командного окна желательно предварительно **сохранить новый вариант файла**. Но можно запускать редактируемый файл на выполнение, не выходя из редактора (т.е. не переходя в командное окно) с помощью пункта меню **Debug/Run**. При этом предварительное сохранение текста исправлений не требуется.

Одной из важных особенностей данного редактора является то, что после проведения вычислений можно в редакторе просмотреть значения переменных, которые они имеют в текущий момент в рабочей области. Для этого достаточно установить курсор мыши на этой переменной, и появится прямоугольник с желтым фоном, на котором выводится текущее значение переменной. Если переменная является большой матрицей, то таким образом увидеть целиком ее не удастся. Для просмотра (и возможного исправления при отладке) всех значений матрицы есть специальный пункт меню **View/Workspace Browser**. Имеется еще два интересных пункта в меню **View**. Это пункт **Evaluate Selection**, который позволяет вычислять значение выделенного выражения и помещать результат в командное окно, и пункт **Auto Indent Selection**, который выполняет автоматическое форматирование текста программы с отступами в соответствии с правилами **MatLab**. Выбор пункта меню **View/Options** позволяет получить доступ к диалоговому окну, в котором можно настроить параметры редактора.

Отладка программного кода - это процесс, в ходе которого могут быть выявлены ошибки двух видов:

- **синтаксические,** которые связаны с неточностью записи имен m-функций или арифметических выражений. **MatLab** обнаруживает большинство синтаксических ошибок, сопровождая их сообщением об ошибке с указанием номера строки соответствующего m-файла;
- **ошибки времени выполнения,** которые, как правило, имеют алгоритмическую природу и проявляются в том, что приводят к непредвиденным результатам.

Достаточно легко можно исправить синтаксические ошибки, которые сопровождаются сообщениями о причинах их возникновения. Ошибки времени выполнения выявить более сложно, потому что локальная рабочая область m-функции оказывается потерянной, если ошибка приводит к возврату в рабочую область системы **MatLab**. Чтобы определить причину такой ошибки, можно использовать один из следующих приемов:

- реализовать вывод результатов промежуточных вычислений на дисплей, удалив в соответствующих операторах точки с запятой, которые подавляют вывод на экран промежуточных результатов;
- добавить в **m-файл** команды **keyboard**, которые останавливают выполнение **m-файла** и разрешают проверить и изменить переменные рабочей области вызываемой m-функции(в этом режиме появляется специальное приглашение), а возврат к выполнению функции реализуется командой **return**);
- закомментировать заголовок функции и выполнить m-файл как сценарий (это позволяет проследить результаты промежуточных вычислений в рабочей области системы);
- использовать отладчик системы **MatLab**.

Отладчик полезен для исправления ошибок во время выполнения программы именно потому, что он дает возможность отслеживать рабочие области функции и проверять или изменять значения соответствующих переменных. Отладчик позволяет устанавливать и удалять контрольные точки, то есть специальным образом помеченные строки **m-файла**, в которых выполнение останавливается. Это дает возможность изменять содержимое рабочей области, просматривать стек вызова m-функций и выполнять **m-файл** построчно. Отладчик может функционировать как в режиме командной строки, так и в режиме графического интерфейса пользователя.

Далее мы рассмотрим отладку только в режиме графического интерфейса пользователя, поскольку он наиболее прост и нагляден. Рассмотрим возможности отладки, которые нам предоставляет **Editor/Debugger**.

Для его запуска используется команда **edit <имя_файла>** или пункт меню **File/Open.** Можно открыть окно редактора/отладчика и с помощью пункта меню **File/New/M-file.** При таком варианте имя отлаживаемого файла открывается уже из меню самого редактора/отладчика. Общий вид окна редактора показан на рис. 5.3.3-2.

Способ использования кнопок понятен из их названия. Часть кнопок представляют собой обычные редактирующие кнопки, используемые при сохранении, копировании, печати и поиске файлов. Другая часть связана непосредственно с отладкой. Это кнопки установки и очистки точек остановки, кнопки пошагового перемещения по программе с заходом в подпрограммы (**Step in**) и без захода в под-
программы (**Single Step**), кнопка продолжения вычислений (**Continue**) и кнопка остановки отладки (**Quit Debugging**). Как было отмечено выше, с помощью пункта меню **View/Workspace Browser** можно не только просмотреть, но и изменить значение любой переменной. При этом в одном из окон редактора открывается таблица, подобная электронной таблице, и в ней можно не только просматривать, но и исправлять значения переменных.

3.5.3.4.4. Редактор m-файлов

В **MatLab** имеется редактор **m-файлов**, для запуска которого следует нажать кнопку **New m-file** на панели инструментов рабочей среды, либо выбрать в меню **File** в команду **New/ m-file**. На экране появляется окно редактора. Наберите в нем какие-либо команды, например для построения графика:

Editor **Пример 5.3.4-1**

```
%
%
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
grid on
title('Экспоненциальная функция')
```

Для запуска программы или ее части есть несколько способов. Первый, самый простой – выделить операторы при помощи мыши, удерживая левую кнопку, или при помощи клавиши **<Shift>** со стрелками, **<PageUp>, <PageDown>** и выбрать **Text** пункт **Evaluate Selection** (или нажать **<F9>**). Выделенные операторы выполняются последовательно, точно так же, как если бы они были набраны в командной строке. Очевидно, что работать в **m**-файле удобнее, чем из командной строки, поскольку можно сохранить программу, добавить операторы, выполнять отдельные команды, не пробегаясь по истории команд, как в случае командной строки.

После того, как программа сохранена в **m-файле**, к примеру, в **myprog.m,** для ее запуска можно использовать пункт **Run** меню **Debug**, либо набрать в командной строке имя **m-файла** (без расширения) и нажать **<Enter>,** то есть выполнить, как команду **MatLab**. При таких способах запуска программы следует учесть важное обстоятельство — путь к каталогу с **m-файлом** должен быть известен **MatLab**. Сделайте каталог с файлом **myprog** текущим.

В **MatLab** в меню **File** рабочей среды перейдите к пункту **Set Path…** Появляется диалоговое окно **Path Browser** (навигатор путей). В строке ввода **Current Directory** установите требуемый каталог. Воспользуйтесь кнопкой, расположенной справа от строки ввода, для выбора каталога.

В **MatLab** установка текущего каталога производится из окна **Current Directory** рабочей среды. Если это окно отсутствует, то следует выбрать пункт **Current Directory** меню **View** рабочей среды. Для выбора желаемого каталога на диске нажмите кнопку, расположенную справа от раскрывающегося списка.

Когда текущий каталог установлен, то все **m-файлы**, находящиеся в нем, могут быть запущены из командной строки, либо из редактора **m-файлов**. Все переменные файл-программ после ее запуска доступны в рабочей среде, т. е. являются глобальными. Убедитесь в этом, выполнив команду **whos.** Более того, файл-программа может использовать переменные рабочей среды. Например, если была введена команда:

Command Window **-Пример 5.3.4-2**

```
>> a=[0.1 0.4 0.3 1.9 3.3];
>>
```

то файл-программа,

Editor -Пример_5_3_4_2

```
%
%
bar(a)
```

содержащая строку **bar(a),** построит столбцевую диаграмму вектора **a** (разумеется, если он не был переопределен в самой файл-программе).

Файл-функции отличаются от файл-программ тем, что они могут иметь входные и выходные аргументы, а все переменные, определенные внутри файл-функции, являются локальными и не видны в рабочей среде. **m-файл**, содержащий файл-функцию, должен начинаться с заголовка, после него записываются операторы **MatLab**. Заголовок состоит из слова function, списка выходных аргументов, имени файла-функций и списка входных аргументов. Аргументы в списках разделяются запятой. Пример 5.3.4-3 содержит простейшей файл-функции с двумя входными и одним выходным аргументами.

Editor -Пример_5_3_4_3

```
function c=mysum(a,b)
c=a+b;
```

Наберите этот пример в новом файле в редакторе и сохраните его. Обратите внимание, что **MatLab** предлагает в качестве имени **m-файла** название файла-функций, т.е. **mysum.m**. Всегда сохраняйте файл-функцию в **m-файле**, имя которого совпадает с именем файл-функции! Убедитесь, что каталог с файлом **mysum.m** является текущим и вызовите файл-функцию **mysum** из командной строки:

Command Window Пример 5.3.4-3

```
>> s=mysum(2,3)
s =
     5
>>
```

При вызове файл-функции **mysum** произошли следующие события:

- входной аргумент **a** получил значение 2;
- входной аргумент **b** стал равен 3;
- сумма **a** и **b** записалась в выходной аргумент **c**;
- значение выходного аргумента **c** получила переменная **s** рабочей среды и результат вывелся в командное окно.

Заметьте, что оператор **c=a+b** в файл-функции **mysum** завершен точкой с запятой для подавления вывода локальной переменной **c** в командное окно. Для просмотра значений локальных переменных при отладке файл-функций, очевидно, не следует подавлять вывод на экран значений требуемых переменных.

Практически все функции **MatLab** являются файл-функциями и хранятся в одноименных **m**-файлах. Функция **sin** допускает два варианта вызова: **sin(x) и y=sin(x)**, в первом случае результат записывается в **ans**, а во втором — в переменную **y**. Наша функция **mysum** ведет себя точно так же. Более того, входными аргументами **mysum** могут быть массивы одинаковых размеров или массив и число.

Разберем теперь, как создать файл-функцию с несколькими выходными аргументами. Список выходных аргументов в заголовке файл-функции заключается в квадратные скобки,

сами аргументы отделяются запятой. В качестве Примера 5.3.4-4 приведена файл-функция **quadeq**, которая по заданным коэффициентам квадратного уравнения находит его корни.

Editor Пример_5_3_4_4

```
function [x1,x2]=quadeq(a,b,c)
%
D=b^2-4*a*c;
x1=(-b+sqrt(D))/(2*a);
x2=(-b-sqrt(D))/(2*a);
```

Command Window Пример 5.3.4-4

```
%При вызове quadeq из командной строки используйте квадратные скобки для
%указания переменных, в которые будут занесены значения корней:
>> [r1,r2]=quadeq(1,3,2)
r1 =
      -1
r2 =
      -2
>>
```

Заметьте, что файл-функцию **quadeq** можно вызвать без выходных аргументов, или только с одним выходным аргументом. В этом случае вернется только первый корень.

Файл-функция может и не иметь входных или выходных аргументов. Заголовки таких файл-функций приведены ниже:

function noout(a,b),
function [v,u]=noin,
function noarg()

Умение писать собственные файл-функции и файл-программы необходимо как при программировании в **MatLab**, так и при решении различных задач средствами **MatLab** (в частности, поиска корней уравнений, интегрирования, оптимизации). Разберем только один пример, связанный с построением графика функции

y=exp(-x).*(sin(x)+0.1*sin(100*pi*x)) на отрезке **[0;1].**

Используйте поэлементные операции для того, чтобы **myfun( )** можно было вызывать от вектора значений аргумента и получать вектор соответствующих значений функции.

Editor Пример_5_3_4_5

```
function y=myfun(x);
%
y=exp(-x).*(sin(x)+0.1*sin(100*pi*x));

%График   y можно получить двумя способами. Первый способ состоит в
%создании вектора %значений аргумента, скажем с шагом 0.01, заполнении
%вектор значений функции и вызове plot:

>> x=[0:0.01:1];
>> y=myfun(x);
>> plot(x,y)
>>
```

Command Window **Пример 5.3.4-5**

```
%%График  y можно получить двумя способами. Первый способ состоит в
%создании вектора значений аргумента, скажем с шагом 0.01, заполнении
%вектор значений функции и вызове plot:
>> x=[0:0.01:1];
>> y=myfun(x);
>> plot(x,y)
>>
```

3.5.3.4.5. Основные операторы m-языка

Помимо программ с **линейной структурой**, инструкции которых исполняются строго по порядку, существует множество программ, структура которых **нелинейна**. При этом ветви программ могут выполняться в зависимости от определенных условий, иногда с конечным числом повторений - циклов, иногда в виде циклов, завершаемых при выполнении заданного условия. Практически любая серьезная программа имеет нелинейную структуру. Для создания таких программ необходимы специальные управляющие структуры. Они имеются в любом языке программирования, и в частности в **MatLab**.

Операторы ввода/вывода. Приведем простой пример диалоговой программы.

Editor **Пример_5_3_5_1**

```
% Вычисление длины окружности с диалоговым вводом радиуса
r=0;
while r>=0,
r=input('Введите радиус окружности r=');
if r>=0 disp(' Длина окружности l='); disp(2*pi*r), end
end
```

Эта программа служит для многократного вычисления длины окружности по вводимому пользователем значению радиуса r. Обратите внимание на то, что здесь мы впервые показываем пример организации простейшего диалога. Он реализован с помощью команды **input:**

input(`'Введите радиус окружности r=');

При выполнении этой команды вначале выводится запрос в виде строки, затем происходит остановка работы программы и ожидается ввод значения радиуса **r** (в общем случае числа). Ввод, как обычно, подтверждается нажатием клавиши <**Enter**>, после чего введенное число присваивается переменной **r**. Следующая строка

if r>=0 disp(' Длина окружности l = '); disp(2*pi*r);end

с помощью команды **disp** при **r>=0** выводит надпись **«Длина окружности 1=»** и вычисленное значение длины окружности. Она представляет собой одну из наиболее простых управляющих структур типа **if...end**. В данном случае она нужна для остановки вычислений, если вводится отрицательное значение **r** (прием, который любят начинающие программисты).

Приведенные строки включены в управляющую структуру **while...end**. Это необходимо для циклического повторения вычислений с вводом значений **r**. Пока **r>=0**. цикл повторяется. Но стоит задать **r<0**, вычисление длины окружности перестает выполняться, а цикл завершается.

Если данная программа записана в виде **m-файла Пример_5_3_5_1**, то работа с ней будет выглядеть следующим образом:

Command Window — Пример 5.3.5-1

```
Введите радиус окружности R=1
Длина окружности l=
6.2832
Введите радиус окружности R=2
Длина окружности l=
12.5664
Введите радиус окружности R=-1
»
```

Функция **input** может использоваться и для ввода произвольных строковых выражений. При этом она задается в следующем виде:

input('Комментарий', V)

При выполнении этой функции она останавливает вычисления и ожидает ввода строкового комментария. После ввода возвращается набранная строка. Это иллюстрирует следующий пример:

Command Window — Пример 5.3.5-2

```
» S=input('Введите выражение ','s')
Введите выражение (Вводим) 2*sin(l)
S =
    2*sin(l)
» eval(S)
ans =
        1.6829
>>
```

Обратите внимание на то, что функция **eval** позволяет вычислить выражение, заданное в символьном виде.

Условный оператор. Условный оператор **if** в общем виде записывается следующим образом:

if *Условие1*
Инструкции_1
else If *Условие2*
Инструкции_2
else
Инструкции_3
end

Эта конструкция допускает несколько частных вариантов. В простейшем

if *Условие*
Инструкции
end

Пока *Условие* возвращает логическое значение **1** (то есть «истина»), выполняются *Инструкции*, составляющие тело структуры **if...end**. При этом оператор **end** указывает на

конец перечня инструкций. Инструкции в списке разделяются оператором **,** (запятая) или **;** (точка с запятой). Если *Условие* не выполняется (дает логическое значение **0**, «ложь»), то *Инструкции* также не выполняются.

Еще одна конструкция

if *Условие*
  Инструкции_1
else
  Инструкции_2
end

выполняет *Инструкции_1*, если выполняется *Условие*, или *Инструкции_2* в противном случае.

Условия записываются в виде:

Выражение_1 Оператор_отношения Выражение_2,

причем в качестве *Операторов_отношения* используются следующие операторы: **==, <, >, <=, >= или ~=.**

Оператор цикла – for...end. Оператор цикла типа **for...end** обычно используются для организации вычислений с заданным числом повторяющихся циклов. Конструкция такого цикла имеет следующий вид:

for var= s:d:e
  Инструкция
  ....
  Инструкция
end

где **s** — начальное значение переменной цикла **var**, **d** — приращение этой переменной и **e** - конечное значение управляющей переменной, при достижении которого цикл завершается. Возможна и запись в виде **s:e** (в этом случае **d=l**). Список выполняемых в цикле инструкций завершается оператором **end**.

Следующие примеры поясняют применение цикла для получения квадратов значений переменной цикла:

Command Window **Пример 5.3.5-3**

```
» for 1=1:5 i^2, end;
ans =
         1
ans =
         4
ans =
         9
ans =
        16
ans =
        25
» for x=0:.25:1 X ^ 2, end:
ans =
         0
ans =
```

```
          0.0625
ans =
          0.2500
ans =
          0.5625
ans =
        1
>>
```

Оператор ***continue*** передает управление в следующую итерацию цикла, пропуская операторы, которые записаны за ним, причем во вложенном цикле он передает управление на следующую итерацию основного цикла. Оператор ***break*** может использоваться для досрочного прерывания выполнения цикла. Как только он встречается в программе, цикл прерывается. Возможны вложенные циклы, например:

Editor **Пример_5_3_5_4**

```
%
for i=1:3
for j=1:3
A(i.j)=i+j;
end
end
```

В результате выполнения этого цикла (файл **Пример_4_3_5_4.m**) формируется матрица **A**:

Command Window **Пример 5.3.5-4**

```
>> Пример4-3-12
%%%
>>A
A =
     2 3 4
     3 4 5
     4 5 6
>>
```

Следует отметить, что формирование матриц с помощью оператора **:** (двоеточие) обычно занимает намного меньше времени, чем с помощью цикла. Однако применение цикла нередко оказывается более наглядным и понятным. **MatLab** допускает использование в качестве переменной цикла массива **A** размера ***m x n***. При этом цикл выполняется столько раз, сколько столбцов в массиве **A**, и на каждом шаге переменная **var** представляет собой вектор, соответствующий текущему столбцу массива **A**:

Command Window **Пример 5.3.5-5**

```
» A=[1 2 3:4 5 6]
A =
     1 2 3
     4 5 6
» for var=A; var, end
var =   1   4
var =   2   5
var=    3   6
>>
```

Оператор цикла - while...end. Оператор цикл типа **while** выполняется до тех пор, пока выполняется **Условие**:

while *Условие*
Инструкции
end

Editor **Пример_5_3_5_6**

```
%
%
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
grid on
title('Экспоненциальная функция')
```

Command Window **Пример 5.3.5-6**

```
%При вызове quadeq из командной строки используйте квадратные скобки для
указания %переменных, в которые будут занесены значения корней:
>> [r1,r2]=quadeq(1,3,2)
r1 =
      -1
r2 =
      -2
>>
```

Досрочное завершение циклов реализуется с помощью операторов ***break*** или ***continue***.

Оператор множественного выбора - switch. Для осуществления множественного выбора (или ветвления) используется конструкция с переключателем типа **switch:**

switch *switch_Выражение*
case *case_Выражение*
Список_инструкций
case {*case_Выражение1,*
case_выражение2, case_Выражение3....}
Список_инструкций
otherwise. *Список_инструкций* **end**

Если выражение после заголовка **switch** имеет значение одного из выражений *case_Выражение...,* то выполняется блок операторов **case**, в противном случае — список инструкций после оператора **otherwise**. При выполнении блока **case** исполняются те списки инструкций, для которых *case_Выражение* совпадает со *switch_Выражением*. Обратите внимание на то, что case_Выражение может быть числом, константой, переменной, вектором ячеек или даже строчной переменной. В последнем случае оператор **case** истинен, если функция **strcmp(***значение, выражение***)** возвращает логическое значение «**истина**».

Поясним применение оператора **switch** на примере **m-файла Пример_5_3_5_7.m**:

Editor Пример_5_3_5_7

```
%Пример 5-3.4-7
switch van
  case {1,2,3}
      disp('Первый квартал')
  case {4,5,6}
      disp('Второй квартал')
  case {7,8,9}
     disp('Третий квартал')
  case {10.11,12}
     disp('Четвертый квартал')
  otherwise
    disp('Ошибка в задании')
end
```

Эта программа в ответ на значения переменной **van** – номера месяца – вычисляет, к какому кварталу относится заданный месяц, и выводит соответствующее сообщение:

Command Window Пример 5.3.5-8

```
>> var=2; swl
Первый квартал
>>var=4;swl
Второй квартал
>> var=7:swl
Третий квартал
>> var=12;swl
Четвертый квартал
>var=-1;swl
Ошибка в задании
>>
```

Для остановки программы используется оператор ***pause***. Он используется в следующих формах:

- **pause** — останавливает вычисления до нажатия любой клавиши;
- **pause(N)** — останавливает вычисления на N секунд;
- **pause on** — включает режим отработки пауз;
- **pause off** — выключает режим отработки пауз.

Следующий пример поясняет применение команды **pause:**

Editor Пример_5_3_5_9

```
%Пример 5.3.4-9
for i=1:20;
x =rand(1,40); y =rand(1,40); z = sin(x.*y);
tri = delaunay(x.y);
trisurf(tri,x,y,z)
pause;
end
```

Команда **pause** обеспечивает показ 20 рисунков - построений трехмерных поверхностей из треугольных окрашенных областей со случайными параметрами.

3.5.4. Контрольные вопросы по теме «Основы работы с математическим пакетом MathCad»

1. Какие основные элементы размещены на экране рабочего окна?
2. Как создать новый документ Mathcad?
3. Каким образом в поле рабочего окна открыть панель Математика?
4. Что входит в состав алфавита входного языка?
5. Какие форматы представления чисел используются в пакете Mathcad?
6. Какие числовые константы имеет пакет Mathcad?
7. Как образуются имена переменных?
8. Что такое встроенная функция?
9. Что такое оператор присваивания и как его вставить в документ?
10. Какое назначение имеет в Mathcad символ = ?
11. Что такое дискретная переменная и как ее задать?
12. Как в Mathcad задать функцию пользователя?
13. Какова последовательность действий для получения таблицы значений функции?
14. Как изменить формат результата?
15. Где расположен шаблон матрицы?
16. Как ввести индекс элементу вектора?
17. Какая встроенная функция позволяет изменить нижнюю границу индекса вектора?
18. Какие арифметические действия можно производить с матрицами?
19. Что требуется ввести, чтобы получить обратную матрицу?
20. Как вычислить определитель матрицы?
21. Можно ли сразу после создания документа начать работу в формульном редакторе?
22. Что такое курсор ввода?
23. Что такое местозаполнитель?
24. Каким образом можно перемещаться между местозаполнителями?
25. Какие действия требуется выполнить, чтобы ввести показатель степени?
26. Как вставить шаблон панели Математика в вводимую формулу?
27. Как выделить часть формулы?
28. Как удалить часть формулы?
29. Как произвести вставку оператора в определенное место формулы?
30. Как ввести в документ текстовый объект?
31. Какие существуют способы выбора шаблона графика?
32. В чем заключается процедура построения графика от одной переменной?
33. Каким образом построить на одном шаблоне два и более графика?
34. Для чего при построении графика требуется использовать дискретную переменную?
35. Как вызывается окно форматирования графика?
36. Какие параметры можно установить с помощью окна форматирования?
37. В чем заключается процедура построения графика поверхности?
38. Можно ли в одном шаблоне построить графики нескольких двумерных функций?
39. В чем заключается процедура построения графика трехмерной диаграммы?
40. Каким образом можно обеспечить вращение трехмерного графика?
41. В чем заключается процедура построения контурного графика?
42. Что такое оператор символьного вывода и на какой панели расположена соответствующая ему кнопка?
43. Какова последовательность действий при разложении функции на множители?
44. Какие средства символьного процессора позволяют упростить математическое выражение?

45. С помощью какого ключевого слова происходить разложение выражения на простые множители?
46. Какой символьный оператор позволяет привести подобные слагаемые?
47. Какую последовательность действий требуется выполнить, чтобы определить полиномиальные коэффициенты заданного выражения?
48. Какая панель инструментов предназначена в Mathcad для решения задач математического анализа?
49. Что требуется ввести для того, чтобы получить числовое значение конечной суммы ряда x^i, если i=1, 2, …10?
50. Какой символ позволяет произвести символьные вычисления бесконечных сумм или бесконечных произведений рядов?
51. Какие средства символьного процессора позволяют вычислять пределы?
52. Что требуется ввести для вычисления значения производной в точке?
53. Как получить аналитическое выражение производной для заданной функции?
54. Как ввести в формулу шаблон определенного интеграла?
55. Как произвести аналитическое вычисление неопределенного интеграла?
56. Можно ли средствами Mathcad проводить вычисления кратных интегралов?
57. Каким образом в текущий документ Mathcad произвести вставку из другого документа?
58. Что требуется выполнить, чтобы некоторую область документа Mathcad экспортировать в другое приложение?
59. Как изменить размеры некоторой области документа?
60. Что требуется выполнить, чтобы изменить цвет выделенной области документа?
61. Что требуется подготовить, чтобы при копировании объекта Mathcad в документ Word он зафиксировался в нужном месте?
62. Что такое текстовый регион и как его задать?
63. Как с использованием элементов панели Форматирования можно произвести форматирование фрагмента текста?
64. Как перед печатью документа произвести его просмотр?
65. Как произвести установки опций страницы с использованием команды меню Файл?
66. Как при печати документа установить количество нужных экземпляров?

3.5.5. Контрольные вопросы по теме «Основы работы с математическим пакетом MatLab»

3.5.6.Тестовые задания по теме «Основы работы с математическим пакетом MathCad»

1. **Система MathCad является**
   1) универсальной математической системой
   2) текстовым редактором
   3) алгоритмическим языком программирования
   4) системой работы с базами данных

2. **К аналогам системы MathCad не относится**
   1) Basic
   2) MatLAB
   3) Mathematica
   4) Maple

3. **Панель Математика содержит**
   1) кнопки с палитрами часто используемых математических обозначений
   2) кнопки с математическими действиями
   3) списки математических функций
   4) кнопки для программирования функций

4. **Символом := в MathCad обозначается**
   1) присваивание
   2) равенство
   3) приближение
   4) описание

5. **Операция присваивания в документе MathCad имеет вид:**
   1) a:=5
   2) a=5
   3) a; 5
   4) a:5

6. **При вводе символа присваивания с клавиатуры, следует ввести**
   1) двоеточие
   2) точку с запятой
   3) пробел
   4) знак равенства

7. **Чтобы изменить формат результата нужно**
   1) изменить количество знаков в окне Формат результата
   2) добавить к результату ноль
   3) изменить точность вычислений
   4) ввести исходные данные с большей точностью

8. **Перед вводом нижнего индекса элемента вектора следует нажать клавишу**
   1) [
   2) Shift
   3) Alt
   4)

9. **Дискретная переменная позволяет**
   1) задать переменной ряд чисел, выстроенных в порядке возрастания с равным шагом
   2) задать переменной интервал изменения
   3) изменить значение переменной на единицу
   4) в списке нет правильного ответа

10. **Встроенные функции, расположенные на палитре Калькулятор вводятся в документ**
   1) щелчком мыши по имени функции
   2) только вводом имени функции с клавиатуры
   3) выделением имени функции на палитре Калькулятор
   4) в списке нет правильного ответа

11. **Если при задании дискретной переменной шаг опущен, то**
   1) шаг считается равным 1
   2)
   3) шаг считается равным 0
   4) шаг считается равным 0.5
   5) Сообщение об ошибке

12. **Числовая константа TOL предназначена для**
   1) задания точности при приближенных вычислениях*
   2) задания количества знаков при выводе результата
   3) задания точности исходных данных
   4) задания количества выводимых результатов

13. **Чтобы получить таблицу значений функции f(x) в заданном дискретной переменной диапазоне следует ввести**
   1) f(x) =
   2) f(x):=
   3) f(x) и нажать клавишу TAB
   4) f(x) ?

14. **Функцию пользователя**
   1) нужно описать до ее первого применения *
   2) можно описать в любом месте документа
   3) можно не описывать
   4) в списке нет правильного ответа

15. **Чтобы получить таблицу значений функции $y(x) = x^2$, для которой значения x заданы дискретной переменной, надо ввести**
   1) y(x)=
   2) y(x):=
   3) $y(x) = x^2$
   4) В списке нет правильного ответа

16. Курсор ввода отмечен

1) красным крестиком
2) горизонтальной линией красного цвета
3) красной вертикальной линией
4) в списке нет правильного ответа

17. Местозаполнитель символа это

1) черный прямоугольник, в который согласно формату должен быть введен символ
2) черная прямоугольная рамка, в которую модно ввести формулу
3) красный прямоугольник, в котором выводится сообщение об ошибке
4) в списке нет правильного ответа

18. Для перехода от одного местозаполнителя к другому можно использовать клавишу

1) TAB
2) HOME
3) END
4) в списке нет правильного ответа

19. Чтобы удалить часть формулы надо

1) выделить эту часть и нажать клавишу Del
2) нажить клавишу Del
3) нажить клавишу Backspace
4) В списке нет правильного ответа

20. Для начала ввода текста непосредственно в вычислительную область следует ввести

1) символ ”
2) символ /
3) символ \
4) символ ?

21. В месте, отмеченном курсором, шаблон графика появляется посредством

1) щелчка мыши по соответствующему шаблону на панели График
2) перетаскивания мышью шаблона графика
3) двойного щелчка по месту, отмеченному курсором
4) в списке нет правильного ответа

22. Если при построении графика аргумент функции не описан, то

1) по умолчанию график будет построен в диапазоне от -10 до 10 с шагом 1
2) по умолчанию график будет построен в диапазоне от -1 до 1 с шагом 0.1
3) появится сообщение об ошибке
4) в списке нет правильного ответа

23. При построении в одном шаблоне 2-х и более графиков имена функций вводятся

1) через запятую
2) через двоеточие
3) через пробел
4) в списке нет правильного ответа

24. Чтобы удалить график надо

1) активизировать его щелчком мыши и нажать клавишу Del
2) нажать клавишу Del
3) выделить график с помощью мыши
4) в списке нет правильного ответа

25. Чтобы изменить графику цвет, символ или толщину линии в окне Форматирования графика используется вкладка

1) Traces
2) X-Y Axes
3) Labels
4) Defaults

26. Символ “->” (стрелка) предназначен

1) для проведения символьных исчислений
2) для ввода комментарий
3) для аналитического преобразования функции
4) для вычисления функции

27. Ключевое слово simplify используется

1) при упрощении выражений
2) при разложении выражения на множители
3) при приведении подобных слагаемых
4) в списке нет правильного ответа

28. Ключевое слово factor используется

1) при разложении выражения на множители
2) при упрощении выражений
3) при приведении подобных слагаемых
4) в списке нет правильного ответа

29. Ключевое слово parfrac используется

1) при определении полиномиальных коэффициентов
2) при упрощении выражений
3) при приведении подобных слагаемых
4) в списке нет правильного ответа

30. Ключевое слово collect используется

1) при приведении подобных слагаемых
2) при определении полиномиальных коэффициентов
3) при упрощении выражений
4) в списке нет правильного ответа

31. Для аналитического решения задач математического анализа предназначена панель

1) Математика
2) Форматирование
3) Стандартная
4) Калькулятор

32. Для выполнения операции подстановки значения переменной в выражение используется ключевое слово

1) substitute
2) complex
3) solve
4) laplace

33. При подстановке значения переменной в выражение после ключевого слова substitute в местозаполнитель следует ввести

1) логическое выражение
2) переменную
3) формулу
4) константу

34. Для вычисления предела $\lim_{x \to \infty} \frac{1}{x}$ используется символ

1) $\to$
2) $=$
3) $\approx$
4) $\div$

35. Для проведения аналитического дифференцирования используется символ

1) $\to$
2) $=$
3) $\approx$
4) $\div$

36. В местозаполнители оператора $\frac{d}{d\blacksquare}\blacksquare$ дифференцирования требуется ввести

1) функцию, зависящую от аргумента и имя аргумента
2) имя функции и имя аргумента
3) имя производной функции и имя аргумента
4) в списке нет правильного ответа

37. В местозаполнители оператора $\int_{\blacksquare}^{\blacksquare} \blacksquare \, d\blacksquare$ егрирования требуется ввести

1) функцию, зависящую от аргумента, имя аргумента и пределы интегрирования
2) имя функции, имя аргумента и пределы интегрирования
3) имя производной функции и имя аргумента
4) в списке нет правильного ответа

38. Для проведения аналитического дифференцирования используется символ

1) $\to$
2) $=$
3) $\approx$
4) в списке нет правильного ответа

39. Если интеграл расходится, то

1) выдается сообщение об ошибке
2) вычисляется расходящийся интеграл
3) интеграл заменяется сходящимся
4) в списке нет правильного ответа

40. Для перемещения элемента оформления документа его нужно

1) выделить и перетащить с использованием мыши
2) скопировать и вставить в нужное место
3) удалить и вставить в нужное место
4) в списке нет правильного ответа

41. Чтобы изменить размеры элемента документа требуется

1) выделить и растянуть (или сжать), потянув за черные прямоугольники на границах области выделения
2) несколько раз черкнуть мышью в пределах области элемента
3) перенести элемент документа в Word и там изменить размеры
4) в списке нет правильного ответа

42. Выделение цветом производится с использованием

1) пункта Свойства элемента меню Формат
2) панели Форматирование
3) пункта Обновить элемента меню Вид
4) в списке нет правильного ответа

43. Чтобы создать текстовый регион требуется ввести символ

1) двойная кавычка
2) двоеточие
3) открывающая квадратная скобка
4) апостроф

44. Для форматирования текста в Mathcad служит

1) панель Форматирование
2) панель Стандартная
3) пункт Формат главного меню
4) в списке нет правильного ответа

45. Для установки абзаца используется

1) маркеры на линейке
2) пункт Формат главного меню
3) панель Форматирование
4) в списке нет правильного ответа

46. Количество копий при печали документа устанавливается в окне

1) в окне печати документа
2) в окне установки опций страницы
3) командой Печать
4) в списке нет правильного ответа

3.5.7.Тестовые задания по теме «Основы работы с математическим пакетом MatLab»

1. **MATLAB – это сокращение от слов**
   1) Mathematical Laboratory (математическая лаборатория)
   2) Matrix Laboratory (матричная лаборатория)
   3) Materialized Labour (овеществленный труд)

2. **Пакеты расширений системы MatLab называются**
   1) Toolkits
   2) Tools
   3) Toolboxes

3. **Из перечисленных устройств не является обязательным при работе с MatLab**
   1) монитор
   2) процессор
   3) принтер

4. **Способна ли система MatLab выполнять операции над комплексными числами**
   1) да
   2) нет

5. **Помимо вызова программ, составленных на языке MatLab, работа в среде MatLab может выполняться**
   1) "в автоматическом режиме"
   2) "в режиме ввода данных"
   3) "в режиме калькулятора"

6. **Большинство команд и функций системы хранится в виде текстовых файлов с расширением**
   1) r
   2) .m
   3) .p

7. **Какое меню в строке меню главного окна MatLab содержит команды для отображения и сокрытия внутренних окон программы**
   1) Window
   2) Help
   3) Desktop

8. **Какое окно системы MatLab предназначено для ввода чисел, переменных, выражений и команд, для просмотра результатов вычислений и отображения текстов программ**
   1) Command History
   2) Command Window
   3) Workspace

9. **Клавиши <↓ > и <↑> в MatLab служат**
   1) для перемещения курсора вниз или вверх по экрану
   2) для перемещения курсора влево или вправо по экрану
   3) для отображения в строке ввода ранее введенных с клавиатуры команд и выражений

10. Если результат вычисления выражения не был присвоен никакой другой переменной, то программа MatLab всегда сохраняет его в переменной

1) inf
2) ans
3) NaN

11. Для отделения целой части числа от дробной в MatLab используется

1) точка
2) запятая
3) точка с запятой

12. Какой формат представления результатов вычислений используется в MatLab по умолчанию

1) hex
2) long
3) short

13. Для обозначения мнимой единицы в комплексных числах в MatLab зарезервировано два символа

1) i и j
2) i n k
3) j и к

14. Требуется ли в MatLab, как и в других языках программирования, заранее декларировать типы переменных

1) да
2) нет

15. Для переноса длинных формул на другую строку используется символ

1) двоеточия
2) точки с запятой
3) многоточия

16. При задании векторов и матриц применяются

1) круглые скобки
2) квадратные скобки
3) фигурные скобки

17. Можно ли при создании матрицы обойтись без символа точки с запятой

1) да
2) нет

18. Какое из утверждений является корректным

1) для вывода нескольких последовательно расположенных элементов вектора используется индексация с помощью оператора двоеточия (:)
2) для вывода конкретного элемента вектора используется индексация с помощью оператора двоеточия (:)
3) для вывода нескольких последовательно расположенных элементов вектора используется индексация с помощью оператора возведения в степень (^)

19. Можно ли с помощью команды save сохранить текст сессии

1) да
2) нет

20. Для построения графиков в линейном масштабе используется функция

1) bar
2) plot
3) subplot

21. Функция loglog служит для установки логарифмического масштаба

1) по оси ординат
2) по оси абсцисс
3) по обеим координатным осям

22. Какая функция позволяет разделить графическое окно MatLab на несколько подо кон и вывести в каждом их них графики различных функций

1) subplot
2) figure
3) plotyy

23. Дополнительный аргумент графических функций plot, semilogx, semilogy, loglog и polar, позволяющий управлять параметрами линий на графике, может состоять максимум из

1) двух символов
2) трех символов
3) четырех символов

24. Какие параметры линии графика задают символы ' yd: ' в дополнительном аргументе графической функции

1) штриховая линия зеленого цвета с маркерами в виде звездочек
2) желтые маркеры в виде крестиков, не соединенные между собой
3) пунктирная линия желтого цвета с маркерами в виде ромбов

25. Для включения линий сетки на графике используется команда

1) grid on
2) grid off

26. Команда text позволяет отобразить

1) надпись в заданном месте графика
2) название горизонтальной оси
3) заголовок графика

27. Программа MatLab сохраняет графическое окно в файле с расширением

1) . fig
2) .mat
3) .doc

28. Для создания матрицы с нулевыми элементами служит встроенная функция

1) null
2) zeros
3) ones

29. Встроенные функции MatLab, позволяющие формировать массивы определенного вида (такие, как zeros, ones, eye и т.д.), могут принимать два аргумента, причем

1) первым аргументом задается число столбцов, а вторым – число строк формируемой матрицы
2) первым аргументом задается число строк, а вторым – число столбцов формируемой матрицы

30. Горизонтальную конкатенацию матриц можно выполнить при условии, что исходные матрицы имеют

1) одинаковое число строк
2) одинаковое число столбцов
3) нулевые элементы

31. Для извлечения строк или столбцов матрицы следует выполнить

1) конкатенацию
2) индексацию с помощью запятой
3) индексацию с помощью двоеточия

32. Если задана некоторая матрица A, то с помощью команды A (end, :) можно

1) извлечь последнюю строку данной матрицы
2) извлечь последний столбец данной матрицы
3) извлечь последний элемент из последней строки этой матрицы

33. Операции поэлементного преобразования векторов могут выполняться

1) только над векторами одинакового размера и типа
2) над векторами произвольного размера и типа
3) только над вектор-строками

34. Какой из перечисленных ниже операторов является оператором поэлементного умножения

1) *
2) .*
3) **

35. Умножение матрицы на матрицу в математике возможно лишь в том случае, когда

1) количество столбцов первого сомножителя равно количеству строк второго соножителя
2) матрицы имеют одинаковые размеры
3) матрицы являются квадратными

36. Длину вектора можно определить с помощью функции

1) dlina
2) width
3) length

37. По умолчанию перемножение элементов массива с помощью функции prod выполняется

1) по столбцам
2) по строкам

38. Для чего используются операторы ".+" и ".-"

1) для выполнения поэлементного сложения и вычитания
2) для сложения и вычитания матриц
3) таких операторов в **MatLab** не существует

39. Среди арифметических операторов наибольший приоритет имеют

1) операторы возведения в степень
2) операторы сложения и вычитания
3) операторы умножения и деления

40. Можно ли использовать операторы отношения для поэлементного сравнения двух матриц

1) да
2) нет

41. Могут ли операторы отношения использоваться в выражениях, вводимых в командном окне системы MatLab, наряду с арифметическими операторами

1) да
2) нет

42. Результатом логической операции "исключающее ИЛИ" будет 1 лишь в том случае

1) когда оба операнда равны нулю
2) когда оба операнда не равны нулю
3) когда один из операндов равен нулю, а другой не равен

43. Какое из утверждений является верным

1) приоритет логических операторов (кроме оператора логического отрицания) ниже, чем приоритет арифметических операторов
2) приоритет логических операторов (кроме оператора логического отрицания) выше, чем приоритет арифметических операторов
3) вычисление выражений всегда происходит слева направо, независимо от приоритета операторов

44. Каким образом нужно задать в MatLab полином, чтобы применить к нему встроенные функции

1) в виде вектора, элементами которого являются корни полинома
2) в виде вектора, элементами которого являются коэффициенты полинома
3) одной переменной присвоить значение степени полинома, а другой — вектор коэффициентов полинома

45. Какое из утверждений является неверным

1) число элементов вектора, задающего коэффициенты полинома, должно быть на единицу больше степени полинома
2) в векторе, задающем коэффициенты полинома, также должны содержаться ну левые коэффициенты
3) в векторе, задающем коэффициенты полинома, можно не указывать нулевые коэффициенты

46. С помощью какой функции в MatLab можно выполнить обращение матрицы

1) с помощью функции inv
2) с помощью функции pinv
3) с помощью функции sinv

47. В каком формате нужно задать функцию eig, чтобы для некоторой матрицы А полу чить матрицу собственных значений и матрицу собственных векторов

1) в формате L=eig (A)
2) в формате [V,E] =eig(A)
3) в формате [V,E,L]=eig(A)

48. В отличие от функций exp, log, sqrt, матричные функции expm, logm, sqrtm

1) выполняют поэлементные операции над матрицами
2) производят вычисления с целыми матрицами по правилам линейной алгебры

49. Для построения трехмерных линий используется функция

1) а) 3plot
2) 6)plot3
3) в) plot33

50. Функция mesh применяется для создания

1) закрашенных поверхностей
2) каркасных поверхностей
3) двухмерных массивов с информацией о координатах узлов сетки прямоугольной области определения, на которой строится трехмерный график

51. Для чего используется команда shading interp

1) чтобы скрыть отображение линий поверхности и сгладить цвета между соседними элементами поверхности
2) только чтобы скрыть отображение линий поверхности
3) чтобы вернуться к параметрам поверхности, заданным по умолчанию

52. Каким образом при построении контурных графиков можно задать программе количество уровней, для которых следует построить изолинии

1) используя функцию contourn, где л – это количество изолиний
2) задав четвертым входным аргументом функций contour и contour3 скалярное значение, соответствующее количеству изолиний

53. Как узнать точные координаты некоторой точки на двух- или трехмерном графике функции

1) отобразить на экране легенду или цветовую палитру
2) на панели инструментов Figure (График) графического окна щелкнуть на кнопке Data Cursor (Указатель данных), а затем щелкнуть на нужной точке графика

54. Каким способом можно прикрепить к определенной точке графика линию, стрелку или надпись

1) с помощью команды Unpin
2) с помощью команды Insert Arrow
3) с помощью команды Pin to Axes

55. Какой формат используется по умолчанию при вводе текста на график

1) формат TeX
2) формат LaTeX
3) ни один из форматов

56. Какое расширение имеют m-файлы в MatLab

1) расширение .mat
2) расширение . т
3) расширение . f

57. М-файлы какого типа могут принимать исходные данные в виде набора входных па раметров и выдавать результаты в виде набора выходных значений

1) файл-программы
2) файл-функции

58. Является ли правильным утверждение, что переменные, определенные в файл-функции, после ее выполнения становятся доступны в рабочем пространстве и могут использоваться в других файл-функциях?

1) да
2) нет

59. Созданный m-файл можно сохранить

1) только в текущем рабочем каталоге
2) в любом каталоге, для которого в **MatLab** установлен путь поиска
3) в любом каталоге, независимо от того, имеется ли он в пути поиска

60. Выберите, какое из следующих утверждений является верным

1) имя m-файла, в котором хранится файл-функция, может совпадать с именем любой переменной или команды **MatLab**, поскольку все переменные, заданные в файл-функции, являются локальными
2) имя m-файла, в котором хранится файл-функция, должно быть уникальным и не должно совпадать с именем функции
3) имя m-файла, в котором хранится файл-функция, должно быть уникальным и должно обязательно совпадать с именем функции

61. Допускается ли вызывать созданную файл-функцию из других файл-программ или файл-функций

1) да
2) нет

62. Какую команду нужно ввести в командное окно, чтобы вызвать редактор m-файлов системы MatLab

1) команду edit
2) команду cd
3) команду pwd

63. Какой цвет по умолчанию использует редактор m-файлов для выделения синтаксических ошибок в коде программы

1) синий
2) красный
3) зеленый

64. Какой символ позволяет обозначить блок программы как отдельный фрагмент

1) символ %
2) символ %%
3) символ %%%

65. Как вызвать диалоговое окно, используемое для установки путей поиска

1) с помощью команды FilePreferences
2) посредством команды FileSet Path
3) командой Filelmport Data

66. Чтобы получить в MatLab максимально возможное значение, представленные в формате с одинарной точностью, нужно ввести в командную строку команду

1) realmax('double')
2) realmin('single')
3) realmax('single')

67. Сколько байтов требуется для хранения каждого элемента массива логических значений

1) 1 байт
2) 2 байта
3) 4 байта

68. Для хранения матрицы с нулевыми элементами в формате sparse

1) требуется больше памяти, чем для хранения матрицы в обычном виде (включая все нулевые и ненулевые элементы)
2) требуется меньше памяти, чем для хранения матрицы в обычном виде (включая все нулевые и ненулевые элементы)

69. Элементом какого массива является элемент к (2 ,2) = {' magic'}

1) массива ячеек
2) массива структур
3) массива символов

70. Какая функция позволяет создать шаблон массива ячеек (массив заданного размера с пустыми ячейками)

1) функция celldisp
2) функция cell
3) struct

71. Если информацию можно представить в виде таблицы с полями, содержащими данные одинакового типа, то для хранения такой информации используют

1) массивы ячеек
2) массивы структур
3) числовые массивы

72. Для удаления ненужного поля в массиве структур используется функция

1) fieldnames
2) getfield
3) rmf ield

73. Какой из перечисленных способов задания символьной переменной является в MatLab ошибочным

1) задание числового кода символа в качестве аргумента функции char
2) ввод нужный символа в апострофах
3) ввод требуемого символа в фигурных скобках

74. Какая функция позволяет выполнить команду, сформированную в виде строки символов

1) eval
2) feval

75. Какци встроенные языки программирования используются в СКМ?

1) Интерпретируемые
2) Компелируемые
3) Низкого уровня
4) Здесь нет правильного ответа

82. Каким образом СКМ решают задачи?

1) Приближенно (численно)
2) Точно (аналитически)
3) Приближенно и точно
4) Здесь нет правильного ответа

83. Где находятся функции и процедуры, предоставляемые СКМ?

1) В ядре
2) В ядре и библиотеках
3) В библиотеках
4) Здесь нет правильного ответа

84. За счет чего обеспечивается кардинальное расширение возможностей СКМ и их адаптация к решаемым конкретным пользователем задачам?

1) Библиотеки
2) Пакетов расширения
3) Справочной системы
4) Здесь нет правильного ответа

85. Чем ограничено наращивание возможностей СКМ?

1) Объемом ядра системы
2) Другими ресурсами СКМ
3) Ничем
4) Здесь нет правильного ответа

86. Какая СКМ лучше всего подходит для символьных вычислений?

1) Mathcad
2) Maple
3) MatLab
4) Derive

87. В какой СКМ удобнее всего работать с массивами?

1) Mathcad
2) Maple
3) MatLab
4) Derive

88. В какой СКМ имеется дескрипторная графика?

1) Mathcad
2) Maple
3) MatLab

89. Модули какого языка прогрммирования позволяет подключать библиотека программного интерфейса MatLab?

1) Pascal и C
2) Fortran
3) C
4) C и Fortran

90. Результаты выполнения каких команд не отображаются в командном окне, даже если после них не стоит ';' ?

1) Команд на построение графиков
2) Команды help
3) Если не ставить ';', то результат выполнения ни одной команды не отображается
4) Здесь нет правильного ответа

91. В какой части главного окна выдаются сообщения об ошибках?

1) В стороке состояния
2) В окне Command History
3) В командном окне
4) Здесь нет правильного ответа

92. Что сохраняется при выполнении команды >>save <название рабочей области>?

1) Все команды, введенные в данном сеансе работы
2) Значения всех переменных данного сеанса работы
3) Все команды и результаты их выполнения
4) Здесь нет правильного ответа

93. В каком виде MatLab представляет все данные?

1) В виде чисел
2) В графическом виде
3) В виде текста
4) В виде массивов

94. Что является главной отличительной особенностью командного окна MatLab?

1) палитры математических знаков
2) неозможность изменить введенную команду, вернувшись на предыдущую строку, после нажатия enter
3) Невозможность задать несколько команд в одной строке

95. Какое расширение имеют файлы, созданные командой save?

1) .mat
2) .txt
3) .m
4) .exe

96. Какая команда используется для освобождения из памяти переменных?

1) Delete
2) Close
3) Clear
4) Здесь нет правильного ответа

97. Для чего используется окно Command Hisory?

1) В нем сохраняются все команды
2) В нем сохраняются только безошибочные команды
3) В нем отмечаются время и дата начала и конца каждого сеанса
4) Здесь нет правильного ответа

98. Как будут отображаться результаты, если выбрать формат long?

1) С 12 цифрами после запятой
2) С 4 цифрами после запятой
3) В экспоненциальной форме
4) Здесь нет правильного ответа

99. Что происходит после команды diary?

1) Все команды и их результаты записываются в текстовый файл
2) Все команды и их результаты сохраняются в окне Command History
3) Все переменные сохраняются в двоичном виде
4) Здесь нет правильного ответа

100. Что происходит с вычислениями если появляется деление на ноль ?

1) Выражения с такой операцией игнорируются
2) Это приводит к ошибке и выходу из программы
3) Данной переменноя присваивается значение inf и выдается предупреждение
4) В зависимости от ситуации программа может вести себя по разному

101. Какой встроенной функцией MatLab вычисляется десятичный логарифм?

1) Lg
2) log10
3) log
4) logarifm

102. Что называется размером массива?

1) Число элементов в массиве
2) Число измерений массива
3) Число элементов вдоль одного измерения
4) Число элементов вдоль каждого из измерений

103. Что приосходит с матрицей после команды c(:,3)=[] ?

1) Удаляется третий столбец
2) Удаляется по три элемента из всех строк
3) Удаляется третья строка
4) Обнуляется третья строка

104. Какая функция используется для создания массива при помощи считывания их из текстового файла?

1) fread
2) dlmread
3) read
4) Здесь нет правильного ответа

105. Что приосходит при выполнении команды A'?

1) Транспонирование A
2) Сопряжение A
3) Нахождение матрицы, обратной к A
4) Здесь нет правильного ответа

106. Для чего в MatLab используется знак \ ?

1) Для деления
2) Для вычитания множеств
3) ля решения систем линейных уравнений
4) Здесь нет правильного ответа

107. Что делает функция max(M,[],1)?

1) Находит максимумы по столбцам
2) Находит максимумы по строкам
3) Находит максимальный элемент матрицы
4) Здесь нет правильного ответа

108. Отличаются ли операции .+ и + ?

1) Да
2) Нет
3) В зависимости от операндов
4) Здесь нет правильного ответа

109. Какое выражение определит скалярное произведение вектор-строк a и b?

1) a*b
2) a.*b
3) a.*b'
4) Здесь нет правильного ответа

110. Для чего нужна среды GUIDE?

1) Для удобства работы с готовыми приложениями
2) Для написания приложений с графическим интерфейсом пользователя
3) Для запуска приложений ToolBox
4) Здесь нет правильного ответа

111. Как в MatLab хранятся приложения с GUI?

1) В файле с расширением .m
2) В файле с расширением .mat
3) В файлах с расширениями .m и .fig
4) Здесь нет правильного ответа

112. К чему приводит добавление элемента интрефейса из редактора приложения?

1) К созданию нового М-файла
2) К созданию нового файла с расширением .fig
3) К созданию соответствующей подфункции
4) Здесь нет правильного ответа

113. Как ускорить решение задачи линейной алгебры, если матрица оказалась разреженной?

1) Все алгоритмы ускоряются автоматически
2) Определить матрицу как разреженную
3) Решение задачи нельзя ускорить
4) Здесь нет правильного ответа

114. Объектами какого класса являются символьные переменные?

1) double array
2) symbolic
3) sym object
4) Здесь нет правильного ответа

115. Чем отличаются функции и операции для работы с символьными переменными ?

1) Перед всеми функциями и операторами ставится sym
2) Для них существует свой набор функций
3) Ничем
4) Здесь нет правильного ответа

116. Какой ToolBox не имеет приложений с графическим интерфейсом?

1) Optimization
2) PDE
3) Все имеют
4) Здесь нет правильного ответа